Improve Java Performace with Better Code

REVIEWS:
- WebGain Studio 4.0 Standard Edition
- JPython 1.1

JAVA Pro™

www.java-pro.com
A Fawcette Technical Publication
December 2000, Vol. 4, No. 12

AVOID PRIORITY INVERSION

Message-Driven BEANS

Maintain Persistent Web Connections

Why You Should Learn to Parse XML

JSP Adds Web Page Flexibility

Free Source Code
Code Online
www.java-pro.com

Enlighten Browser Users

U.S. $5.95 (Canada $7.95)

01>

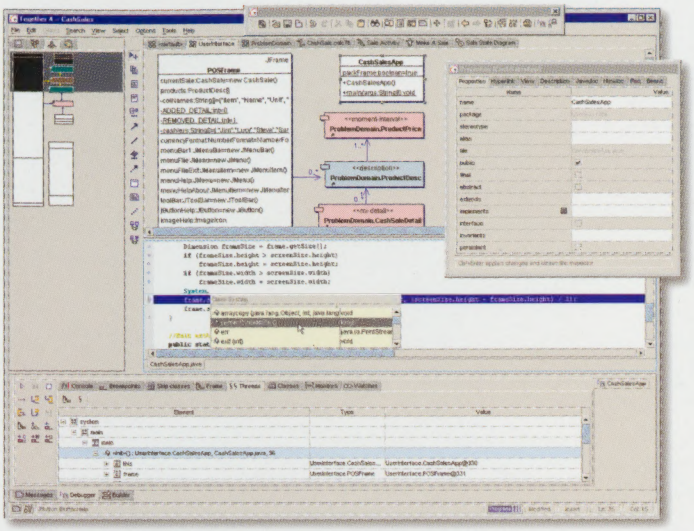
7 25274 91607 8

model → edit → gen doc → version → audit/metric → compile → debug → provision → assemble → deploy → run → model → edit → gen doc → version → audit/metric → compile → debug → provision → assemble → deploy → run → model → edit →

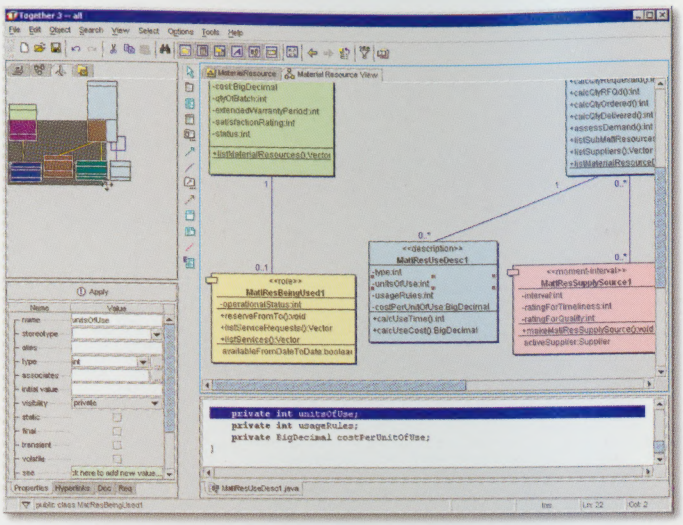
New! Together Control Center 4.1

Model • Pattern • Edit • Gen Doc • Version • Audit/Metric • Compile • Debug • Provision • Assemble • Deploy • Run

"the backbone of all our software development" —*The Home Depot*



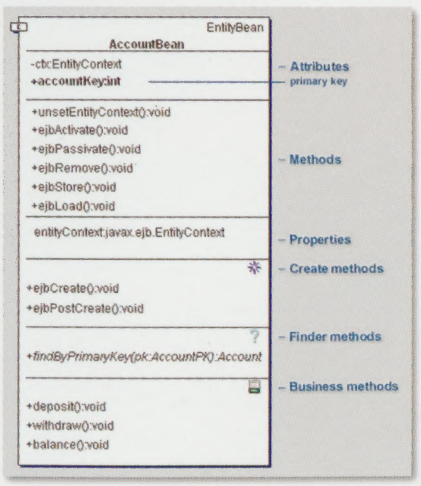
OneSource™ *simultaneous round-trip engineering!* Class diagrams and source code, always in-sync, always up-to-date, giving you complete freedom to work visually or textually (Java, C++ or IDL) as you see fit. Plus robust programming editor with syntax highlighting and auto-completion features; compiler with messages auto-linked to diagram element, source-code line, and property editor; and distributed, multi-threaded debugger.

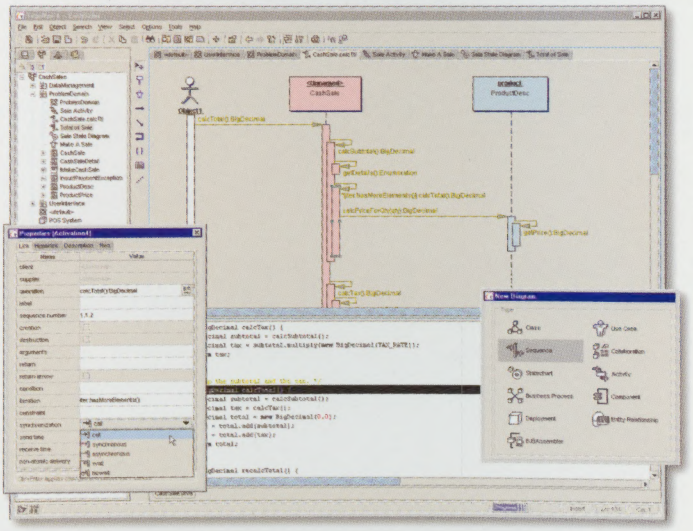


BigPlay™ *team support is just a mouse-click away!* Complete end-to-end multi-user version control for models, source and documentation across the enterprise (using CVS or whatever SCC-compliant version control system you prefer).

EJB features:

- Generate, configure, reconfigure, customize, and compile
- Visually edit your create, finder and business methods in any EJB
- Together auto-updates the supporting interfaces and primary-key classes (hidden from view, unless you choose otherwise)
- Round trip EJB-JDBC, ERD-JDBC, and JSP-JavaBean support
- Generate, configure, reconfigure and customize GoF patterns
- Add your own company-specific patterns (express simple patterns in text, complex in Java)





UML 1.3 *Nine kinds of diagrams!* Reverse engineer any source code into sequence diagrams, ERDs, too. Plus Raftmaker™ customizable multi-level doc (Javadoc 2 superset + RTF) can be invoked as part of your daily build. Deliver always up-to-date doc to your boss!

Download ***Together Whiteboard*** at **www.togethersoft.com/jp**

"Having seen Together's direct use of Java code to support the models, I am forced to wonder why anyone would like to work in any other way! ...it's a complete solution, not just a modeling tool." —**Jon Collins, Senior Analyst, Bloor Research**

"We discovered in our evaluation that other modeling tools only get used during the front end of a project. Yet we found Together to be the one UML modeler that our developers will use from start to finish. It's the only one that delivers simultaneous round-trip engineering (rather than batch mode). Why?" —**Curtis Chambers, The Home Depot**

"Thanks for your wonderful product! I wonder how you guys find the time to incorporate so many additional new features in the short span of time between 3.0 and 4.0. I am, as a power user of your product, very happy to see your progress." — **Suresh Balamaran, TimeO**

TogetherSoft™

Together™ Control Center™ Vendor Independence Platform™

TogetherSoft Corporation
920 Main Campus Drive, Suite 410, Raleigh, NC 27606 USA
919-833-5550

©2000 TogetherSoft Corporation, all rights reserved. Not all features are available in all editions.
Together, OneSource, BigPlay, and Raftmaker are trademarks of TogetherSoft Corporation. Java, JavaBean, and EJB are trademarks of Sun Microsystems, Inc.

model → edit → gen doc → version → audit/metric → compile → debug → provision → assemble → deploy → run → model → edit → gen doc → version → audit/metric → compile → debug → provision → assemble → deploy → run → model → edit →

© Copyright 2000 by Sitraka Inc. All rights reserved. Sitraka, the Sitraka logo and JClass are trademarks of Sitraka Inc. and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. All other products are trademarks or registered trademarks of their respective companies.

"The difference between delivering a good Java™ app and a great one!"

ComputerWire OO Tools Bulletin

JClass™

The world's leading Java components.

JClass Chart

JClass LiveTable

JClass PageLayout

JClass Field

JClass HiGrid

JClass DataSource

JClass JarMaster

JClass Elements

Turn your applications into great ones with JClass, the world's leading suite of Java components. Proven in enterprise environments around the globe, JClass Enterprise Suite includes all the functionality you need to build powerful and great-looking interfaces: charts, grids, pop-up calendars, masked edit fields, GUI extensions, JAR optimization, printing and reporting, data connectivity and more. Add to that unrivaled JDK, platform and IDE support, and you're looking at one exciting solution.

Download a free eval now

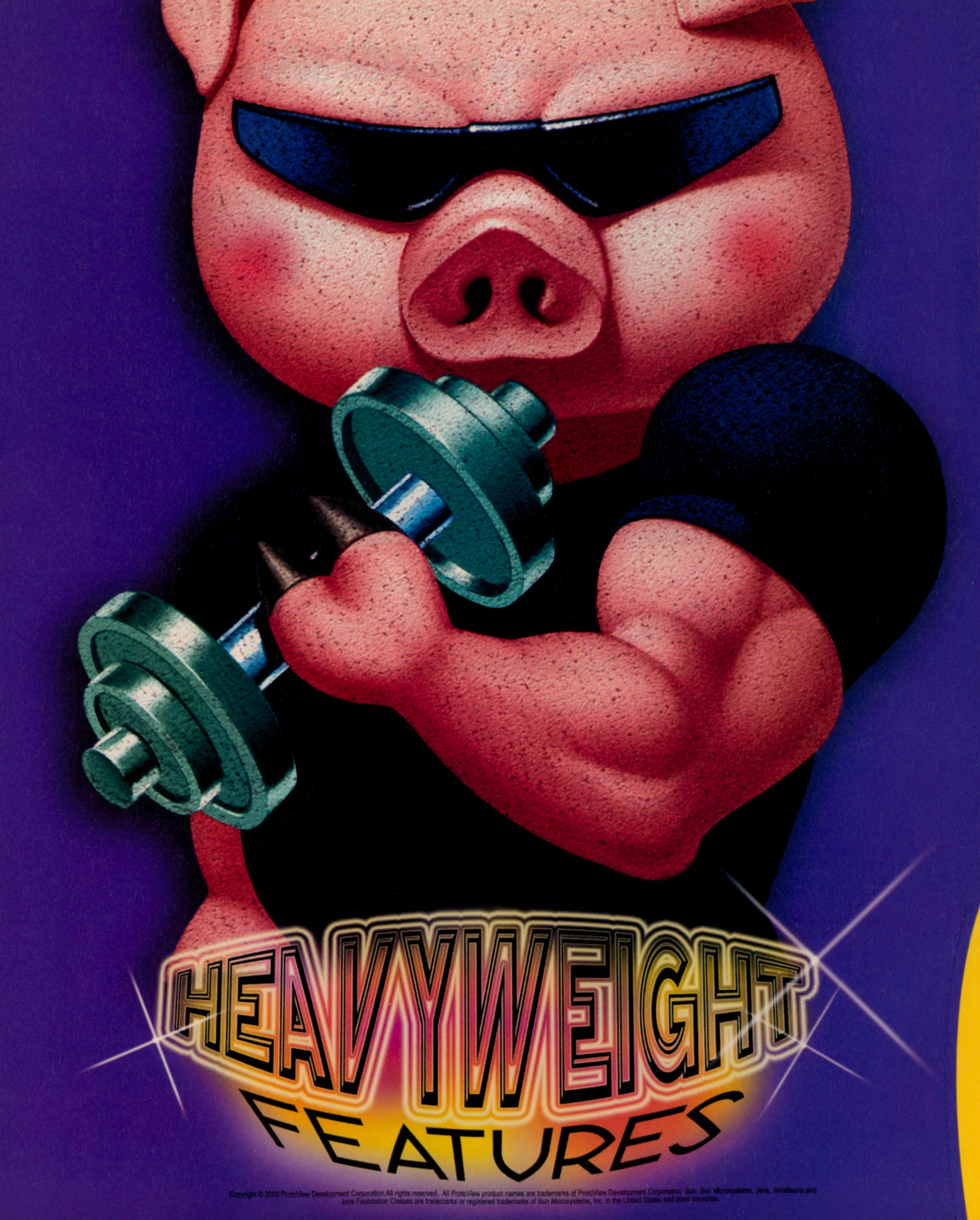
www.sitraka.com/greatapp

Formerly KL Group 1 888-361-3264 | Europe +31(0)20 510 67 00

sitraka

software

the Java™ advantage



Copyright © 2000 ProtoView Development Corporation.All rights reserved. All ProtoView product names are trademarks of ProtoView Development Corporation. Sun, Sun Microsystems, Java, JavaBeans and Java Foundation Classes are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

JFCSuite & JSuite

Put more muscle behind your applications with the ProtoView JFCSuite and JSuite. The JFCSuite adds capabilities to the Swing JTable that you may not have thought possible: column locking, vertical and horizontal cell merging, sorting, in-cell validation, printing and much more.

PowerChart (included in both suites) makes charts look fantastic with instantaneous data binding from any source, server-side rendering and vivid 2D & 3D rendering.

Both the JFCSuite for Java2™ (Swing) and JSuite for AWT, have everything you need to create your own magic: grid, tree, charting, calendaring and rich graphical data input components.

With the smart data models and rich presentation features of JFCSuite and JSuite, you'll be firming up in no time.

PowerChart

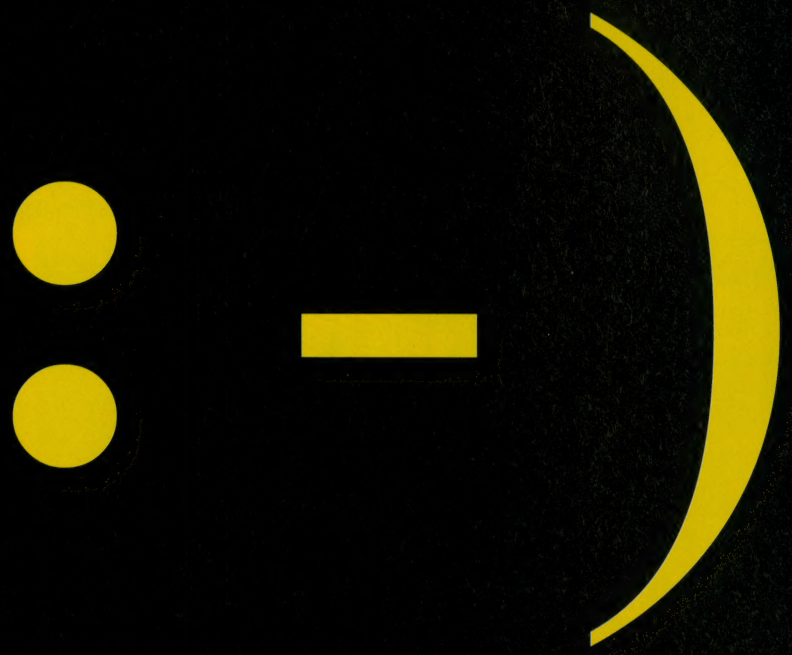
PowerChart brings Java charting to a whole new level. Instantaneous data binding from any source with dynamic update capabilities. Support for server-side rendering* and thin client delivery of chart images. Interactive, real-time chart editing and updating. Vivid 2D & 3D rendering with smooth zoom and rotate. Detailed control of legend, appearance, attributes and behaviors. Support for all major chart types. Check out PowerChart and watch your data pump some iron!

For complete details and trial versions, visit: www.protoview.com

Order online or call 800-231-8588

ProtoView
www.protoview.com

* Requires J2EE



Have a nice update.

PowerUpdate™ by Zero G.™

Smile. Updating just got a lot easier.

Introducing PowerUpdate, the easiest live updating solution ever.

PowerUpdate keeps your users' software updated by automatically notifying them when there is a new version, and then downloading and installing it. PowerUpdate can be added to virtually any software application, uses any existing deployment solution, and runs on Windows, Solaris or Linux. And it's free. Finally, software updating that will make you smile!

Get happy! www.ZeroG.com/update

©2000 Zero G Software, Inc. PowerUpdate, InstallAnywhere and Zero G are trademarks of Zero G Software, Inc.
All other trademarks are property of their respective owners.

zero G
ZERO G

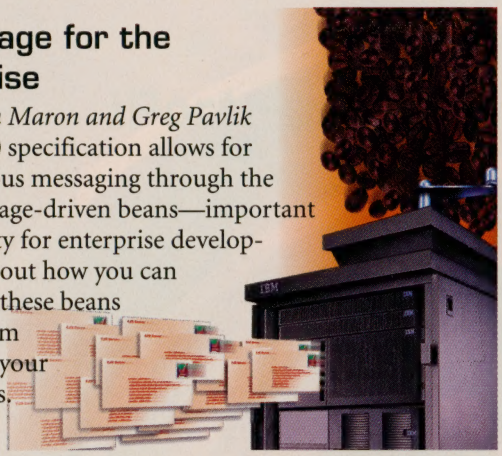
JAVA Pro™

January 2001, Vol. 5, No. 1

Contents

Features

20 A Message for the Enterprise
by Jonathan Maron and Greg Pavlik
The EJB 2.0 specification allows for asynchronous messaging through the use of message-driven beans—important functionality for enterprise development. Find out how you can implement these beans and use them to enhance your applications.



34 Improve Java Performance
by Tarak Modi
If you think that Java performance is poor, it might be your coding that's at fault. Learn some tricks for optimizing your programs and see if you get better results.

40 The Payoff of Persistence
by Cheng-Yaw Chang
Connecting multiple Web clients to legacy systems presents problems when connections are not serializable, and reopening and reauthenticating clients adds expensive overhead. See how two scalable Web architectures use connection objects to maintain persistent connections.

44 Send Users a Browser Message
by Timothy Eden
Many unsavvy browser users can't tell that an application is still processing. If your audience is like this, you can use the servlet session API and JavaScript to send them a message.

Reviews

12 WebGain Studio 4.0 Standard Edition
Product suite for Java Web app development

16 JPython 1.1
An object-oriented, 100% Pure Java, open source programming language

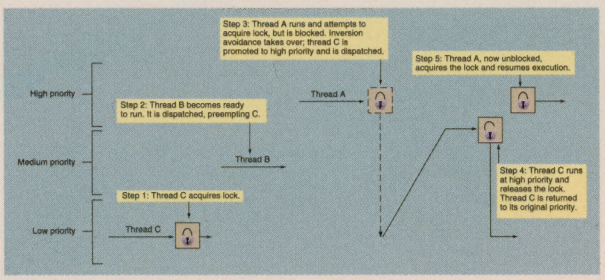
Columns

51 Javatecture
by James Cooper
JavaServer Pages can be used for more than just controlling the data you display on your browser—they can also control how the data is displayed. Find out how.

60 Visual Components
by Claude Duguay
Swing provides a powerful architecture, but selecting elements from a JTree could be easier. Here's how you can develop a flexible and easy-to-use JTree with checkbox selections.

66 Java To Go
by Rick Grehan
Avoiding priority inversion is critical for the proper operation of real-time systems. Find out what solutions are available.

74 Pro Shop
by Daniel F. Savarese
Parsing XML can have a high overhead. Still, XML can be applied broadly in Java applications, and SOAP may establish it as the underlying data format for distributed computing. Daniel discusses how it works.


Step 3: Thread A runs and attempts to acquire lock, but is blocked. Inversion avoidance takes over; thread C is promoted to high priority and is dispatched.
Step 5: Thread A, now unblocked, acquires the lock and resumes execution.
High priority
Step 2: Thread B becomes ready to run. It is dispatched, preempting C.
Thread A
Thread B
Medium priority
Step 4: Thread C runs at high priority and releases the lock. Thread C is returned to its original priority.
Step 1: Thread C acquires lock.
Low priority
Thread C

Avoid priority inversion; see page **66**.

Departments

9 Editor's Note
by Steve Gillmor

18 New Products

79 Ad Index

80 Public Static
by Sean Gallagher

BPA INTERNATIONAL

PRINTED IN USA

Java Pro is published monthly—plus a special issue in September—for $34.97 per year, by Fawcette Technical Publications Inc., 209 Hamilton Avenue, Palo Alto, CA, USA, 94301-2500. Tel. 650-833-7100; Fax. 650-321-3818. **Postmaster:** Send address changes to *Java Pro*, P.O. Box 54584, Boulder, CO 80322-4584. Application to mail at Periodicals Postage Rates is pending at Palo Alto, CA and additional mailing offices. **© 2001 Fawcette Technical Publications Inc.**, all rights reserved. *Java Pro* is an independent publication not affiliated with Sun Microsystems. Sun Microsystems is not responsible in any way for the editorial policy or other contents of the publication. All contents of *Java Pro* are copyright © 1999 by Fawcette Technical Publications Inc., unless otherwise noted. "VBITS" and "Interactive Developer" are trademarks of Fawcette Technical Publications Inc., a California Corporation, James E. Fawcette, President. Java is a trademark of Sun Microsystems. Rather than put a trademark symbol in every occurrence of other trademarked names, we state that we are using the names only in an editorial fashion with no intention of infringement of the trademark. Although all reasonable attempts are made to ensure accuracy, the publisher does not assume any liability for errors or omissions anywhere in the publication. **Canadian GST#: 86831 0764 ISSN: 1096-4495.** Ride-Along and Standard Mail Enclosed Presorted Standard US Postage Paid, Permit #17, Waseca, MN 56093.

Visit IONA's website, www.iona.com/products/ip_ipas_home.htm to learn more about iPortal Application Server. Interested in learning about highly scalable, flexible and dynamic application structure? Register for a FREE webcast at www.iona.com/pswebcastsQ101

© IONA Technologies PLC. iPortal Application Server and Xtreme Scalability are trademarks of IONA Technologies PLC. All other product, service, and company names are trademarks or registered trademarks of their respective owners.

005900C23

IONA®

iPortal Application Server™

Bringing *Xtreme Scalability*™ to EJB

Searching for an application server to...

- Provide full interoperability with existing applications?
- Provide a deployment environment with unparalleled flexibility, scalability, and performance?
- Support an *enterprise portal* or a similarly demanding large-scale e-business application?

Your search ends here

The IONA iPortal Application Server is a J2EE-based environment for developing and assembling large-scale e-business systems. It offers a uniquely powerful graphical environment for building and managing EJB applications.

Visit IONA's website at **www.iona.com** to learn more about IONA's iPortal Application Server.

www.iona.com

The Enterprise Portal Company™

Unrelenting Performance

Only FioranoMQ uses a file-based data store to deliver guaranteed messages significantly faster than any other JMS implementation!

FioranoMQ - Message Server

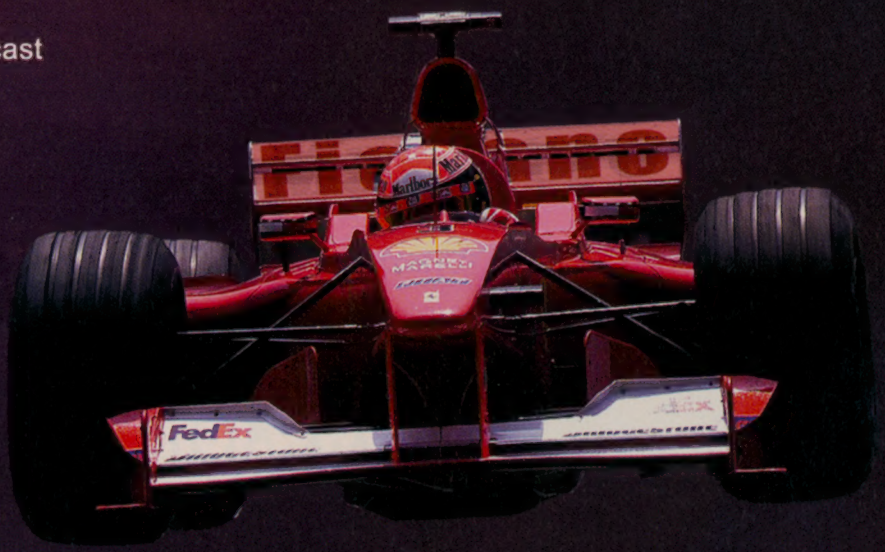
- Fully JMS Compliant, Scalable & Secure
- HTTP Tunneling, XML Interoperability
- Clustering and Failover Protection
- 10x faster than comparable offerings
- Bridges to IBM MQSeries™, MSMQ™
- C++ Interoperability

FioranoMQ™

Internet Messaging Server

FioranoMQ - Info Bus

- Serverless JMS
- Fault Tolerance, High Availability
- Zero Maintenance/Administration
- Reliable Message Delivery over IP Multicast



Fiorano

eBusiness Infrastructure Solutions

Download Free Developer License!

www.fiorano.com, (800) 663-3621, info@fiorano.com

Copyright (c) 2000, Fiorano Software, Inc. All Rights Reserved. Fiorano and FioranoMQ are trademarks or registered trademarks of Fiorano Software, Inc. All other trademarks are property of their respective owners.

Open the Pod Bay Door, HAL

by Steve Gillmor
Editor in Chief
sgillmor@fawcette.com

Now that 2001 is upon us, it's instructive to look back on Stanley Kubrick's 1968 classic and see how it measures up to the real thing. Kubrick's elegant and precise use of technology looks as modern today as it looked futuristic when first rendered on the silver screen.

Science fiction has always been a subtle blend of recycled sociology and predictive behavioralism. "2001" both reflected and influenced the space race: Kubrick's designs for the EVA backpacks were emulated by NASA, and Boeing's flight avionics user interfaces were copied for the Pan Am space shuttle cockpit sequences.

The film's central theme concerns the relationship between man and machine, and it questions what happens when computers grow so "intelligent" that they develop what many call a soul. In today's globally wired world, peer-to-peer technologies are being proposed that would harness the processing power of multiple desktop machines to emulate (or surpass) the power of supercomputers such as the HAL 9000 of the movie.

The rapid evolution of processor speed and miniaturization that began with the space program has brought many of HAL's core technologies to the marketplace at commodity prices. Videoconferencing and motion detection software let users and machines "read lips." Moore's Law is pushing natural language processing and speech recognition down to desktop office suites and embedded systems.

Asynchronous messaging constructs such as awareness and file-system sharing utilities like Napster are spreading virally as Web services, treating the network as one large storage or relay device. HAL's intelligence was apparently stored onboard in a multi-terabyte server farm, a hint of things to come in today's distributed Web.

The Internet's economy of scale has made virtual communications possible on a much wider scale, but at the cost of security, reliability, and scalability. Collaborating across domains of space and time requires protocols, development languages, and component models that can survive and prosper in the asynchronous world of TCP/IP.

As Daniel Saverese notes ("Pro Shop," page 74), technologies such as XML have emerged to allow applications to communicate and exchange data independently of their native platforms. And as Jonathan Maron and Greg Pavlik detail in our cover story ("A Message for the Enterprise," page 20), Java's ability to process asynchronous messages has only recently become available with J2EE's Java Messaging Service (JMS).

Combine XML's loosely coupled efficiencies with JMS and EJB 2.0's message-driven beans and you get critical services—persistence, verification, and transaction support—deployed in the rich EJB container infrastructure. The same component delivery vehicle that J2EE developers use for RPC-based session and entity beans now supports asynchronous services. And message beans allow concurrent processing, enabling higher throughput and scalability.

These capabilities come none too soon to the Java community in a competitive environment where business technologists will adopt the right combination of rapid development, iterative deployment, and interoperability. Microsoft's evolutionary .Net and IBM's service-oriented architecture both rely on SOAP-driven Web services. And Ray Ozzie's peer-to-peer Groove platform uses SOAP and COM-based message queuing to route XML-encapsulated data between nodes, adding security, awareness, and virtual file sharing services.

HAL 9000's intelligence was apparently stored onboard in a multi-terabyte server farm, a hint of things to come in today's distributed Web

But Groove's most significant contribution may be its transparent success at crossing domain boundaries. The secret weapon: relay servers that store and forward queued messages and broker connections across firewalls between man and machines. Users can create and invite others into secure shared spaces that survive time zones, variable bandwidth, disconnected usage, and corporate directory infrastructures—all without the intervention or acquiescence of IT. No more "Sorry, Dave, I can't do that."

Go Online

Use the following DevX Locator+ code at www.java-pro.com to go directly to this related resource.

JP0101EN_T DevX Premier Club members can read this feature online.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Introducing the New **ONLINE SEMINAR SERIES**

GO inside J2EE

and join a special group of industry leaders.

INSIDE J2EE Online Seminar Schedule

(ALL SEMINARS ARE SCHEDULED FOR 1:00 PM EST)

12/12 Introduction to Enterprise JavaBeans™

The EJB™ server-side component model simplifies development of middleware components that are transactional, scalable and portable. This online seminar will introduce you to the fundamental concepts behind EJB.™

12/14 Integrating SOAP and J2EE

This online seminar will provide an introduction to SOAP document structures, an example of how a J2EE-based XML integration server (SilverStream xCommerce) can be used to process SOAP-formatted message exchanges, and a discussion of J2EE deployment alternatives for SOAP-based integrations.

Archived Seminars

View one at www.silverstream.com/seminararchives

- Deploying J2EE Apps.
- J2EE Performance Tips and Techniques
- WAP/WML Programming for App. Servers
- An Insider's Look at the J2EE Certification Process
- Java Server Pages (JSP)™ Application Design
- Inside Java Message Service (JMS)™

Attend the INSIDE J2EE online seminar series and learn from industry experts how best to build Java™2 Platform Enterprise Edition (J2EE) Web applications. These 45-minute seminars are for anyone just learning about J2EE, developing J2EE applications, or interested in the future of J2EE. Co-sponsored by Sun Microsystems and SilverStream Software, INSIDE J2EE outlines how and when J2EE technologies are best used, the impact of the platform and how it will best deploy your Web applications.

To register, visit www.silverstream.com/onlineseminars or call 1-888-823-9700 (U.S.) or 1-978-840-1765 (outside U.S.).

Register now and enter to win a Palm VII.

Winner will be chosen from attendees and notified at the conclusion of each seminar.

SilverStream
Are you ready to do eBusiness?

Sun microsystems

©2000 SilverStream Software, Inc. SilverStream is a registered trademark of SilverStream Software, Inc. Sun, Sun Microsystems, the Sun logo, Enterprise JavaBeans, Java Server Pages and Java Message Service are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

JAVA Pro

Jeff Miller, *Vice President, Publishing*

Editorial

Sean Gallagher, *Editorial Director*
Steve Gillmor, *Editor in Chief*
Kay Keppler, *Executive Editor*
Janaya Reitz, *Managing Editor*
Charles Ferrara, *Assistant Editor*

Contributors

James W. Cooper, Claude Duguay
Rick Grehan, Daniel F. Savarese

Art & Production

Michael Hollister, *Vice President, Art & Production*
Bruce Gardner, *Senior Art Director*
Christine Marquez, *Production Director*
Kathleen Sweeney, *Associate Production Manager*
Michael Furukawa, *Associate Production Manager*
Casey Mallough, *Associate Art Director*
Khari Helae, *Production Artist*
Kim Le, *Senior Designer*
Marine Leroux, *Web Producer*
Julie Ranahan, *Associate Producer*
Ricardo Villarroel, *Associate Producer*
Hao Luu, *Web Associate*

Advertising Sales

Shana Deane, *Advertising Director*
Cecile Pendleton, *Account Manager*
David Lyman, *Eastern Regional Sales Manager*
Janice Mahler, *Western Regional Sales Manager*
Beatrice Stonebanks, *Southern Regional Sales Manager*
Jean Anne Mahler, *Eastern Regional Manager, CareerLink*
Kathy Bazan, *Western Regional Manager, CareerLink*
Tami Stoeckle, *Production Coordinator*

Circulation

Meg Clark, *Circulation Manager*
Peter Klehm, *Circulation Director*
Evelyn Adenau, *Associate Circulation Director*
Gerry Guzman, *Circulation Associate*
Joyce Hoelzle, *Circulation Associate*

Marketing

Libby Dickinson, *Marketing Communications Manager*
Margaret Horoszko, *Marketing Designer*
Ray Long, *Marketing Production Coordinator*
Susan Ogren, *Marketing Specialist*
Helga Taylor, *Designer*

Conferences

Tena Carter, *Vice President*
Robert Scoble, *Editor, VBITS*
Lee Thé, *Editorial Director*
Maryam Ghaemmaghami, *Associate Editor*
Jennifer Brucker, *Conference Marketing Manager*
Thea Schwartz, *Conference Marketing Director*
Nicole Simonich, *Conferences Marketing Planner*
Shannon Sullivan, *Conference Marketing Planner*
Sarah Lemonds, *Marketing Editorial Assistant*
Cedric Fellows, *Associate Sales Director*
Leslie Hasvold, *Conference Operations Planner*
Janet Nickels, *Conference Operations Manager*
Suzanne Nahlik, *Conference Planner*

Customer Service

David Seymour, *Customer Service Manager*
Audra Tate, *Customer Service Manager*
Melanie Meinert, *Customer Service Representative*
Laurie Groenewold, *Customer Service Representative*

Operations

Philip Ziman, *Director of Human Resources and Facilities*
Aaron Wuele, *IS Manager*
Linda Brooks, *Accounting Operations Manager*
Darlyn Philips, *Director of Finance*
Staci Matlack, *A/P Accountant*
P.J. Van Hoy, *A/R Accountant*
Nancy Burgoa, *Staff Accountant*
Iain Neillands, *Staff Accountant*
Leah Spain, *Accounting Assistant*
Neema Buruku, *Financial Analyst*
Pamela Davis, *HR Manager*
Nancy Wilson, *Office Operations Manager*
Jamie Po'oi, *Facilities Coordinator*
Wendy Gray, *Receptionist*

Fawcette Technical Publications

James E. Fawcette, *President*
Patricia Cameron-Loyd, *Chief Operations Officer*
Jeff Miller, *Vice President, Publishing*
Lynne Matthes, *Group Publisher*
Paul Spyksma, *Publisher, FTP Advertising Network*
Chris Kinsman, *Vice President, Technology*
Peter Horan, *Chief Executive Officer, DevX*
Greg Stern, *Vice President/General Manager, DevX*
Wilson, Sonsini, Goodrich & Rosati, *Corporate Counsel*

ONLINE RESOURCES

Locator+ code: JP0101

Type this code in the Locator+ field at the top of the *Java Pro* home page. You can find all the code in the current issue of *Java Pro* at www.java-pro.com. All the listings and associated files discussed in the articles are available free in one ZIP file. To access code in issues from 1998 and 1999, type "JP" followed by the month and year. For example, the Locator+ code for the October 1998 issue is JP1098. To access code from issues published after December 1999, type "JP" followed by the year (two digits) and the month (two digits). For example, the Locator+ code for the January 2001 issue is JP0101.

Premier Club Members

Members of the Premier Club of the Development Exchange (www.devx.com, $24.95 for six months, $44.95 for one year) can get each article's listings and associated files in separate ZIP files. For example, to download the individual code file associated with the October 1998 "Ask the Java Pro" column by Daniel F. Savarese, you would type in JP1098AP as the Locator+ code. For any articles published after December 1999, however, the position of the year (two digits) and the month (two digits) will be switched in the Locator+ code. For example, to download the individual code file associated with the January 2001 "Pro Shop" column, you would type in JP0101PS. Premier Club members can also access archives of all articles and related code ever published in *Java Pro*.

Java Zone

The Java Zone (www.java-zone.com) on the Development Exchange (www.devx.com) offers additional Java resources including product reviews, programming tips, weekly downloads, and discussion forums.

HOW TO REACH US

Editorial Offices:

Fawcette Technical Publications
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Phone: 650-833-7100
Fax: 650-853-0230
Web: www.java-pro.com
E-mail: java-pro@fawcette.com

Article Submission

To submit articles for publication please contact:
Kay Keppler, Executive Editor, *Java Pro*
E-mail: kkeppler@fawcette.com
Download the Author Guidelines at www.java-pro.com/authguid.asp

Product Reviews

To submit products for review please contact:
Chris Preimesberger, Senior Editor
E-mail: cpreimesberger@devx.com

Product Announcements

Please e-mail press releases to:
pressreleases@fawcette.com

Reprints and Permissions

For all *Java Pro* editorial and advertising reprints contact:
PARS International Corp.
114 East 32nd Street, Suite 503
New York, NY 10016
Phone: 212-221-9595
Fax: 212-221-9195

To quote from an article, please contact:

Tami Stoeckle
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Phone: 650-833-7118
E-mail: tstoeckle@fawcette.com
Specify the issue date and title of the article, the portion you would like to quote, and the purpose.

Photocopy Rights:

Permission to photocopy for internal or personal use may be granted by Fawcette Technical Publications. Please contact Tami Stoeckle for more information.

Customer Service and Subscription Information

For subscription orders, inquiries, or address changes please contact Customer Service:
Java Pro
P.O. Box 54584
Boulder, CO 80322-4584
Phone: 303-945-5282
Fax: 650-321-3818
Foreign and Canadian orders must be payable in U.S. dollars, plus postage. The surface rate to Canada and Mexico is $52.97 per year. For all other countries the air mail rate is $78.97 per year.

Back Issues

To order *Java Pro* back issues, call 650-833-7100 or 800-848-5523 and ask for Customer Service. Back issues cost $10. Additional postage will be charged to deliveries outside the USA.

Reviews

WebGain Studio 4.0 Standard Edition
JPython 1.1

WebGain Studio 4.0 Standard Edition

Advantage through integration?

by Michiel de Bruijn

Quick Facts:

Provided you like the Unified Modeling Language and don't already use alternatives for parts of the suite, WebGain Studio offers a comprehensive solution for serious Java Web application development.

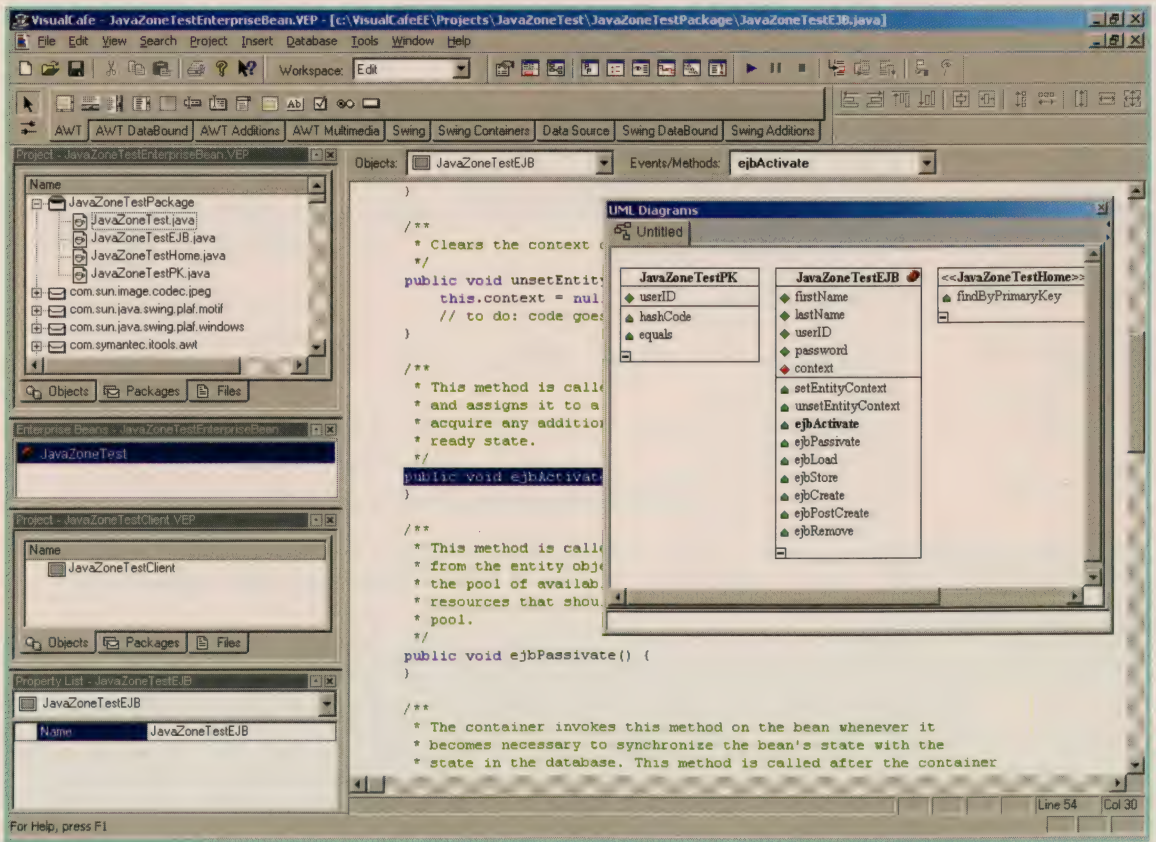
WebGain, Inc.
20300 Stevens Creek Blvd., Suite 200
Cupertino, CA 95014
Phone: 888-822-3409
Web: www.webgain.com
Price: $4,995

Even if you haven't heard of WebGain Studio before, you'll definitely be familiar with the individual components that make up this suite: VisualCafé, which WebGain acquired from Symantec late last year; StructureBuilder, previously from Tendril software; the WebLogic Server, licensed from BEA Systems; and Dreamweaver, contributed to the mix by Macromedia.

VisualCafé, of which the 4.0 Enterprise release is included in WebGain Studio, is still uncontested as the premier Java IDE on the market. It has continually improved on its pioneering features, such as two-way code editing (where changes you make in your visual form design are automatically reflected in your code, and vice versa), transparent switching across multiple JDK versions, and incremental debugging.

Version 4.0 continues that tradition with EJB 1.1 deployment wizards, persistence modeling, and an integrated copy of StructureBuilder, which allows construction of Java classes from Unified Modeling Language diagrams, as well as reverse-engineering, where diagrams are constructed from existing code (see figure). In addition to class diagrams, sequence and use-case diagrams also are supported.

To make it easier to deploy the JavaBeans you develop for Web sites, WebGain supplies you with Dreamweaver. This excellent visual HTML/JavaScript/XML editor offers site management, precise control over just about every visual aspect of your Web pages, and a useful extensibility model that has already resulted in an impressive amount of third-party add-ins.



Paint Me A Picture StructureBuilder, the Universal Modeling Language tool integrated into the VisualCafé environment, allows you to lay the groundwork for Java applications in a visual way, as well as create class diagrams for existing code.

Detecting the Wrong Platform

One such plug-in, which integrates JavaServer Pages (JSP) development and deployment in the main Dreamweaver environment, is provided as part of WebGain Studio. Unfortunately, these extensions don't install properly from the main Studio setup program on Windows 2000 (it detects this platform as Windows NT, and subsequently claims to need a higher service pack than is available for Windows 2000). You'll have to find and rerun the JSP extensions setup program manually afterwards to get it to work.

To ensure enterprise-level scaling for the components you develop, WebGain Studio comes with a single-user development copy of version 5.1 of the WebLogic Java 2 Enterprise Edition (J2EE) transaction server. VisualCafé comes with several wizards intended to make J2EE development easier and includes support for easily deploying JavaBeans to the WebLogic server.

Despite this integration, I had quite a problem getting the highly touted "hot deployment" (updating components while the transaction server is running) to work at all. I also wasn't impressed by the lack of

Ericsson | Echelon | Perkin Elmer | IntelliCorp | KLA-Tencor | BuildNet | Evolve | Lucent | Siemens | US West | IDS Scheer

Looking for a database infrastructure with superior performance and total cost of ownership?

let's talk.

If you're serious about developing eCommerce solutions, don't waste your breath talking to anyone else about Internet infrastructure. Chances are, they'll offer you a piecemeal approach based on the object technology we invented in the first place. So come straight to the source. Where our innovative solutions to meet any Internet challenge are more than just idle chitchat. They're what the rest of the industry is just talking about. Call us at 800-950-8845 or visit our web site at www.poet.com/oss

POET

www.poet.com

WebLogic-specific functionality where component management and debugging are concerned.

Apart from the WebLogic Server, WebGain Studio also works with the latest release of the iPlanet (ex-Netscape) application server. This is a step in the right direction, but it would really help if WebGain also supported application servers that are not as expensive and don't come with as many deployment issues as these two.

Best on the Back End

So far, WebGain Studio offers an impressive collection of products. But why put all these together in one package? Perhaps the answer is that Webgain had to do *something* with the technology it has acquired from its strategic partners.

Fortunately, there is sensible reasoning behind the collection. First, any enterprise Web project needs a serious back end, which is the job of WebLogic server. WebLogic runs JavaBeans developed with a combination of VisualCafé and StructureBuilder. Those components will need to be integrated into production Web sites. A team that will be more comfortable with the Dreamweaver environment and the JSP plug-ins—rather than the Java developers—typically does this. Integration between the products increases productivity and reduces the need for in-house scripting and hacks to get things working.

So far, so good. However, the level of integration offered by WebGain is not exactly convincing, mainly because it's limited to some small areas of integration between individual products and doesn't extend across the entire development process very well.

For WebGain studio to realize its potential fully, I'd expect to see at least some end-to-end site templates to deliver on the "easy e-commerce" promise made by the marketing literature. Without such templates, delivering complete projects with WebGain Studio is not significantly easier than using any other tool collection.

It's likely that your organization already has selected alternatives for one or more of the products included in the suite. Although you can purchase most components of WebGain Studio individually, the Dreamweaver JSP extensions are available only as part of the bundle.

However, if you're starting with a clean slate, WebGain Studio offers a comprehensive answer to the question, "How do I get started with team development of a serious Java-based distributed Web site?" In that case, it would definitely be a good idea to take a close look at this product.

Editor's note: A free upgrade to Version 4.1 is now available for VisualCafé 4.0 Expert Edition owners; see www.webgain.com for details and to register.

Michiel de Bruijn lives in Rotterdam, The Netherlands, and spends his days as a networking and development specialist for an international media group. He seems to enjoy living on the bleeding edge of technology. Michiel always welcomes questions and comments; reach him at mdb@x42.net.

How to Build a Digital Dashboard
Hands-on Technical Solutions
Exchange & Outlook
Migrating to
Exchange 2000
What You Need to Know Now
Master
Exchange 2000!
With the First and Foremost
Exchange Authority
Exchange and Outlook is the ONLY independent Exchange and Outlook magazine, written by the experts exclusively for IT professionals, administrators, managers and corporate developers.
Subscribe online
http://exchange.devx.com/in.asp
Call today 1-800-848-5523 and pay only
$39.95 for a full year: twelve issues in all.
One year is based on twelve issues. Annual cover price is $83.40. Please allow four-to-six weeks for delivery of first issue. International subscriptions must be paid in U.S. dollars plus postage. Canada/Mexico, add $18/year; other countries add $44/year for airmail. Canadian GST included.
Exchange & Outlook magazine is another high-quality, how-to technical publication published by Fawcette Technical Publication. Exchange and Outlook are trademarks of Microsoft Corp. All other product names herein are the properties of their respective owners.
Save 52% off
the cover price!

Programmer's Paradise®

JAMMIN' WITH JAVA

programmersparadise.com • U.S. 800-445-7899 • Canada 888-423-2700

Programmer's Paradise offers a tremendous selection of Java products including those from:

- Borland/INPRISE
- Compuware NuMega
- IBM Corporation
- InstallShield
- Metrowerks
- ProtoView
- Sitraka
- VisualSoft
- WebGain

...and many more!

Call us for all your Java needs—we special order too!

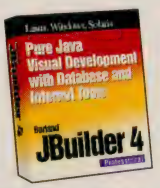
GUARANTEED BEST PRICES*

Should you see one of these products listed at a lower price in another ad in this magazine, CALL US! We'll match the price, and still offer our same quality service and support!

*Terms of the offer:

- Offer good through January 31, 2001
- Applicable to pricing on current versions of software listed
- January issue prices only
- Offer does not apply towards obvious errors in competitors' ads
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.



JBuilder 4 Professional

by Borland/INPRISE

Paradise No.
B19 0753-AU

$540.99

WebGain Studio 4

by WebGain

Paradise No.
W51 0115-AU

$4,989.95

FREE Catalog!

100+ pages of products and information for Developers!

Get a FREE Subscription to our catalog by calling **800-445-7899**

FREE e-mail Newsletter at **programmersparadise.com**

We Stock Hundreds of Java Tools!

New JRun Studio now available!

JRun Server 3.0 Professional

by Allaire Corporation

Allaire® JRun® is an easy-to-use Java® application server and integrated development environment for building and deploying server-side Java applications. With its modular design, JRun lets you choose the J2EE technologies you require in your Java applications, including: EJB 1.1; JTA 1.0; JMS 1.0; JSP 1.1; and Java Servlets 2.2.

Paradise No.
A27 0A20-AU

$768.99

JSmartGrid™ 1.0

by Eliad Technologies, Inc.

Add Excel-like user interfaces to your Java applications by using the smallest Swing grid component (162 KB) designed for Java 2 v1.3.0. The JSmartGrid extensive API with its scalable presentation abilities simplifies design for Web, database and B2B applications. It gives developers more flexibility in displaying all types of data, including XML documents.

Paradise No.
E85 0110 E-HP*

$179.00

* This product is only available for electronic download at www.programmersparadise.com

Sybase Java Enterprise Bundle for NT

by Sybase

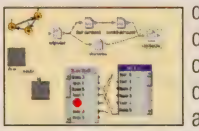
Sybase is pleased to provide this bundle for NT comprising all the tools necessary to build enterprise-class Java applications, from creation, to testing and debugging, to deployment.

The bundle includes: PowerJ; EAServer–Enterprise Application Server; Adaptive Server Enterprise database server; and PowerBuilder® ASE is available for both NT and Unix for genuine multi-platform scalability.

Paradise No.
S84 0226-AU

$1,999.99

New Version! 3.0



JGO Diagram Class Lib 3.0

by Northwoods Software Corporation

With JGO create: network diagrams; process flow diagrams; data flow diagrams; family trees; flow charts; schematics; wiring diagrams; visual languages; and more! We don't just build components, we also build graphical applications. So our components aren't just a guess at what a programmer needs, but are built to meet our requirements and those of our customers.

Paradise No.
N49 0130-AU

$615.99

Call for pricing.

Prices range from $90.99 to $1,420.99

JDBC Drivers

by Atinav, Inc.

The Database Developers' Connectivity Tool of Choice

Atinav provides JDBC Drivers capable of accessing databases on multiple platforms by supporting DBF, MS Access and SQL Server. To address the stringent demands of eBusiness, the Connection Optimized version and the Client Resources Optimized Version (without memory contention) simplify the development process.

Atinav aXeConnect™

Tremendous Selection • Guaranteed Best Prices • Personal Service

programmersparadise.com • U.S. 800-445-7899 • Canada 888-423-2700

JPython 1.1

An object-oriented, 100% Pure Java language

by Peter Varhol

Quick Facts:

This implementation of the object-oriented Python language integrates with the Java platform and is certified as 100% Pure Java. It is free and distributed with source code.

JPython
Web: www.python.org
Price: Free for commercial and noncommercial use

Python is an interpreted, object-oriented programming language that has been ported to several platforms. The language has an elegant syntax, a small number of powerful, high-level data types are built in, and it's easy to learn for most programmers. Python can be extended systematically by adding new modules implemented in a compiled language such as C or C++. Such extension modules can define new functions and variables as well as new object types.

JPython is an implementation of Python written in 100% Pure Java (certified by Sun). It is seamlessly integrated with the Java platform and thus allows you to run Python on any Java platform. JPython interprets Python code down to Java byte code, where it can be executed by the underlying Java virtual machine (see listing).

You can download JPython at www.python.org. It is smaller than a megabyte and comes in two forms: with or without the OROMatcher regular expression library. Most people will want it with the ORO-Matcher library, but the license isn't an open source one and may limit what you can do with the software.

Once you download it, you open a command line and invoke your JVM to unpack and install it. JPython comes with most standard Python modules, and you can also work with Java modules.

I'm building a set of Python objects and methods that act as producers and consumers, to model systems with limited capacities. All of this Python code seemed to execute correctly under JPython. I attempted to make rough performance comparisons manually and could find no discernable difference between interpreted Python and byte code-translated JPython. I don't know if this is generally true or simply an artifact of my code.

I've attempted to use JPython on both Windows 98 SE, the QNX multimedia operating system, and Windows 2000. I encountered problems using the Microsoft JVM, JView, under Windows 98; JPython seemed to install correctly, then presented me with an installation script error. This problem was documented on the JPython Web site, although the site maintainers claim they couldn't replicate it. Their recommended solution was to upgrade to Internet Explorer 5.5.

As this description suggests, running JPython is possible on a wide variety of systems. Certainly JPython is within the reach of any common (and many not-so-common) systems, including the growing variety of embedded systems supporting JVMs. This could potentially make JPython a popular programming language for Internet devices, CGI, and other Web-centric uses.

JPython includes a server-side implementation, Python Server Pages (PSP). PSP is a Java server-side scripting engine designed along the lines of Microsoft's Active Server Pages and Sun's JavaServer Pages specification. JPython also provides several ways of working with relational databases, including the Python language's DB-API. Of course, you can always call a Java method to invoke JDBC and work through Java.

JPython has several commercial uses, primarily as an integrated scripting engine. The Data Subnet Manager, MCI Worldcom's internal frame relay configuration and provisioning suite, uses JPython for regression testing, interactive debugging, and rapid prototyping of its Java client. ObjectDomain, a UML modeling tool, uses JPython as a vehicle for extensions and third party add-ins. The ICEBrowser Java Web browser from ICESoft (now a part of Wind River Systems) includes support for JPython as a scripting alternative to JavaScript.

Why use JPython when you can simply write your programs in Java? The languages are similar, and there aren't any obvious performance advantages in yet another interpreted language. I can think of two reasons. First, you don't have to cope with the license restrictions found in Java. (CNRI holds a copyright on Python, but the language is governed by a variant of an open source license.) You can change the language or extend it to suit your needs better.

Second, Python is a much smaller language than Java. Of course, it lacks many of the features of the Java language and platform, but some programmers simply won't need those features. If that's the case, and if an application's memory footprint is an important issue (as it often is with embedded systems), then JPython may find a role in applications where Java is too big and too inflexible. ■

Peter Varhol is a technical evangelist for Compuware Corporation. You can reach him at Peter.Varhol@compuware.com.

Listing 1 A Command Line Interface JPython lets you use Java classes to create user interfaces, network connections, or anything else you can do with Java. The following code calls the AWT from within a Python function to create a user interface with buttons.

```
from java import awt, applet

class ButtonDemo(applet.Applet):
  def init(self):
    self.b1 = awt.Button('Disable middle button',
      actionPerformed=self.disable)
    self.b2 = awt.Button('Middle button')
    self.b3 = awt.Button('Enable middle button', enabled=0,
      actionPerformed=self.enable)

    self.add(self.b1)
    self.add(self.b2)
    self.add(self.b3)

  def enable(self, event):
    self.b1.enabled = self.b2.enabled = 1
    self.b3.enabled = 0

  def disable(self, event):
    self.b1.enabled = self.b2.enabled = 0
    self.b3.enabled = 1
```

What did one distributed application say to the other?
We need to talk.
Progress SonicMQ™
The world's fastest E-Business Messaging Server
Performance. Not Promises.
© Copyright 2001 Progress Software Corporation. All rights reserved. Progress and all other names of Progress Software Corporation products are registered trademarks or trademarks of Progress Software Corporation. All other trademarks and registered trademarks, whether marked or not, are property of their respective owners.

PROGRESS SOFTWARE

Are your distributed applications on speaking terms? Now there's a solution that guarantees the reliable exchange of information: Progress® SonicMQ™. This 100% Java, fully JMS-compliant, Internet messaging server includes everything you need to enable event-based communication between business-critical applications — whether you use a point-to-point or broadcast model. You can even send XML formatted messages. Need to know more about this standalone, scalable solution? **Call 1-800-989-3773, ext. 325, mail the attached reply card, or visit www.sonicmq.com/jp101.htm for a FREE Developer's CD and information.** Then let's talk.

SonicMQ: Get the message!

Sun microsystems

New Products

Graphics and Charting

Testing and Debugging

Database and Middleware Tools

Integrated Development Environments

Graphics and Charting

Ulead JavaRazor 1.0

Ulead JavaRazor 1.0 is a tool for adding Java buttons and animation to your Web page. It consists of two programs: Ulead Button.Applet and Ulead Animation.Applet. Java Animation.Applet allows users to create rotating, spinning, and dancing applets, while Button.Applet lets users create buttons of any size, style, or shape, with creative and customized rollover effects. $69.95.

Ulead Systems, Inc.
Phone: 310-523-9393; 800-85-ULEAD
Web: www.webutilities.com

JLOOX 1.3

JLOOX 1.3 is a dynamic graphics and data visualization tool that lets Java developers create visualization applications that build upon the capabilities of JFC/Swing technology. This latest version offers improvements in network map visualization, custom graphical editor creation, and integration with legacy data and popular development tools, such as Borland JBuilder, IBM Visual Age, and Sun Forte for Java. $3950.

LOOX Software
Phone: 650-552-9310; 800-684-LOOX
E-mail: info@loox.com
Web: www.loox.com

Testing and Debugging

EJB-test 2.2

EJB-test 2.2 conducts functionality and performance testing of Enterprise JavaBeans' middle-tier applications. It tests EJB components throughout the development cycle, as well as after deployment. New features include advanced session bean support, including XML data support, dynamic graphing capabilities, and extended functional testing, as well as the ability to define a baseline against which subsequent tests can be run and compared. Starts at $20,000.

RSW Software, Inc.
Phone: 781-993-8500
E-mail: info@rswsoftware.com
Web: www.rswsoftware.com

Optimizeit 4.0

Optimizeit 4.0 is a Java profiling tool designed to test and improve the performance of most Java programs: servlets, Enterprise JavaBeans, JavaServer Pages, and complex server-based applications. It pinpoints performance problems such as bottlenecks, excessive object allocations, memory leaks, and inefficient algorithms or processor use. New features include universal Java virtual machine support, offline profiling for testing programs in production environments, and integration with a wider base of application servers. Starts at $499.

Intuitive Systems Inc.
Phone: 408-245-8540
E-mail: optimizeit@intuisys.com
Web: www.optimizeit.com

Test Mentor, Java Edition

Test Mentor, Java Edition is a functional test and test modeling tool, integrated with the VisualAge for Java development environment to make it easier for developers to test as they code. It provides semantics, a visual vocabulary for modeling and documenting tests, shortcuts, wizards, automatic test code generation, and a user interface for evaluating created test elements. It also features an additional user interface for navigating and analyzing test results, which are generated as XML for later analysis or baseline comparison. Starts at $935.

SilverMark, Inc.
Phone: 919-858-8300; 888-588-0668
E-mail: info@silvermark.com
Web: www.silvermark.com

Database and Middleware Tools

HiT Allora for Java

HiT Allora for Java is a database-focused DOM/SAX implementation that helps maintain an XML-consistent interface across all relational databases by allowing both read and write database integration. Developers can use Sun standard JAXP interfaces to build W3C DOM objects, transform them as necessary, update node values, and apply the XML data changes back into the underlying DBMS. HiT Allora for Java supports JDK version 1.2 and later. Available in Java and native Windows versions, and compatible with all major Web application development platforms including BEA Weblogic, IBM WebSphere, iPlanet, SilverStream, and others. Contact for pricing.

HiT Software Inc.
Phone: 408-345-4001
E-mail: info@hit.com
Web: www.hit.com

FioranoMQ Message Server 4.5

FioranoMQ Message Server 4.5 is a JMS-compliant Java messaging server. Delivers scalable, secure, event-driven distributed systems. Uses a file-based data store to deliver guaranteed messages. Supports all the documented Publish/Subscribe and PTP (Point-to-Point) functionality in the JMS API, also adding a highly configurable security system, unmatched scalability, Web integration, and superior XML interoperability. Contact for pricing.

Fiorano Software, Inc.
Phone: 408-354-3210
E-mail: info@fiorano.com
Web: www.fiorano.com

Integrated Development Environments

WebGain VisualCafé 4 Enterprise Edition

WebGain VisualCafé 4 Enterprise Edition provides an environment for developing, debugging, and deployment of Enterprise JavaBeans (EJBs) across J2EE application servers. Re-entrant wizards streamline the development of Enterprise JavaBeans (EJB), with new support for the EJB 1.1 specification. A Persistence Builder allows developers to build persistence models for EJB components, while wizards simplify the deployment of EJBs to leading application servers hosted on various platforms including HP/UX, Sun Solaris, and Windows NT. Single-view distributed debugging makes it possible to view multiple processes running simultaneously on different virtual machines, platforms, and operating systems. VisualCafé supports popular object request brokers, middleware, and databases. $2995.

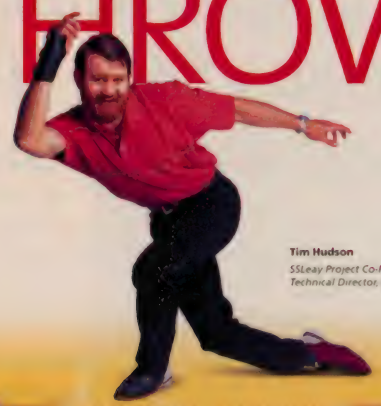
WebGain, Inc.
Phone: 408-517-3700; 877-932-4246
E-mail: sales@webgain.com
Web: www.webgain.com

www.rsasecurity.com/go/bowl

Act now and get a FREE SDK and Special Professional Services Package.

PKI-ENABLE YOUR APPS WITH ONE THROW

See RSA Security products in the MGM movie "AntiTRUST" In theaters January 12, 2001. www.rsasecurity.com/go/mgm



Tim Hudson
SSLeay Project Co-Founder
Technical Director, RSA Security Australia

With **RSA BSAFE Cert-C & J** SDKs

Don't let the need to write and test a bunch of PKI APIs stand in the way of your application development. Bowl 'em over with RSA BSAFE Cert-C and Cert-J e-security components. They're the strong, interoperable solutions for PKI readiness. Write just once and you've got a perfect strike against all your API needs, leaving you with plenty of time to spare. Fast. Simple. Secure. Just what you'd expect from RSA Security, your source for authentication, encryption, and PKI. **1.800.782.5453**

RSA SECURED

Products displaying this symbol contain or are compatible with the most trusted e-security technology

RSA SECURITY

The Most Trusted Name in e-Security

© 2000 RSA Security Inc. All rights reserved. RSA, BSAFE, the RSA Secured logo, "The Most Trusted Name in e-Security" and the RSA Security logo are registered trademarks or trademarks of RSA Security Inc. All other trademarks are the property of their respective owners.

A Message for the Enterprise

Asynchronous messaging—essential in enterprise-class applications—is available with message-driven beans

by Jonathan Maron and Greg Pavlik

The Enterprise JavaBeans (EJB) specification has, up to the 1.1 release, defined two component types—session and entity beans. Client objects invoke methods on both of the EJB 1.1 bean types synchronously. However, with the importance of legacy Message Oriented Middleware (MOM) and the release of several Java Message Service (JMS) implementations, a facility for leveraging the benefits of asynchronous messaging is now also required within the EJB framework. This has been addressed in the EJB 2.0 specification, now in public draft, which defines a third component type—message-driven beans.

Jonathan Maron is a senior architect at Bluestone Software, Inc. on the Total-e-Server development team. He was the principle architect of the Bluestone EJB 1.1 server and is a member of the Javasoft EJB Expert's Group that defined the EJB 1.1 and EJB 2.0 specifications. Jon can be reached at jmaron@bluestone.com.

Greg Pavlik is a senior architect at Bluestone Software, Inc. on the Total-e-Server development team and has contributed to the design and development of the EJB 1.1 and EJB 2.0 servers. Greg can be reached at gpavlik@bluestone.com.

Jon and Greg developed Bluestone's implementation of the EJB 2.0 message-driven bean framework.

Message-driven beans combine the functionalities of EJB and JMS. Of course, message-enabling technology is accessible with the JMS, but this new component type offers new possibilities for messaging. How can these beans be integrated into an application server framework? And how can their usefulness be expanded past the scope of JMS-server implementations? Let's find out.

EJB and JMS

As we mentioned, the EJB 1.1 specification defines two enterprise bean types for developers—session and entity beans. Session beans generally implement some business logic and are not shared among multiple clients. Entity beans represent an object-oriented view of an entity that exists within some persistent storage facility such as a database. In both of these component models, the interaction with the client is facilitated with home and remote interfaces. These interactions are, by definition, strictly synchronous. For example, a request is sent to the bean by a blocking (and potentially remote) method invocation, and the server object returns a response (see Figure 1).

In the scope of an enterprise application, however, there often is a need for nonblocking, asynchronous messages, as well. For example, a client may wish to send the server a message for which no response is expected or needed; in this case, there is no reason for the client to have to wait for the server object to process the request. It is more efficient for the client application to submit a message and continue processing with the assurance that the message eventually will reach the server and be processed appropriately.

Java technology with the ability to process asynchronous messages has only recently become available in the form of the Java Message Service (JMS), which was developed originally to provide a standard Java interface to legacy Message Oriented Middleware (MOM) products from established vendors like IBM and Tibco. These products are important parts of existing systems in many enterprise environments.

Today, companies like Fiorano, Progress, Softwired, and SpiritSoft are marketing a whole new generation of lightweight and efficient 100% Pure Java JMS implementations. These products enable developers to establish JMS connections to publish and receive messages from other application components. Listing 1 shows the typical steps required for interfacing to a JMS provider.

In addition to going through all the steps to obtain the reference to the JMS MessageConsumer, a developer typically creates and registers one or more implementations of the JMS MessageListener interface with the MessageConsumer. A MessageListener is always executed in a single thread of control, which means that the developer does not have to worry about concurrency issues when writing a MessageListener implementation (see Figure 2). Listing 2 shows a typical JMS MessageListener implementation. In this implementation, the messages received by the MessageListener contain stock quotes. The listener simply obtains the stock quote from the message body and prints it out to the standard output stream.

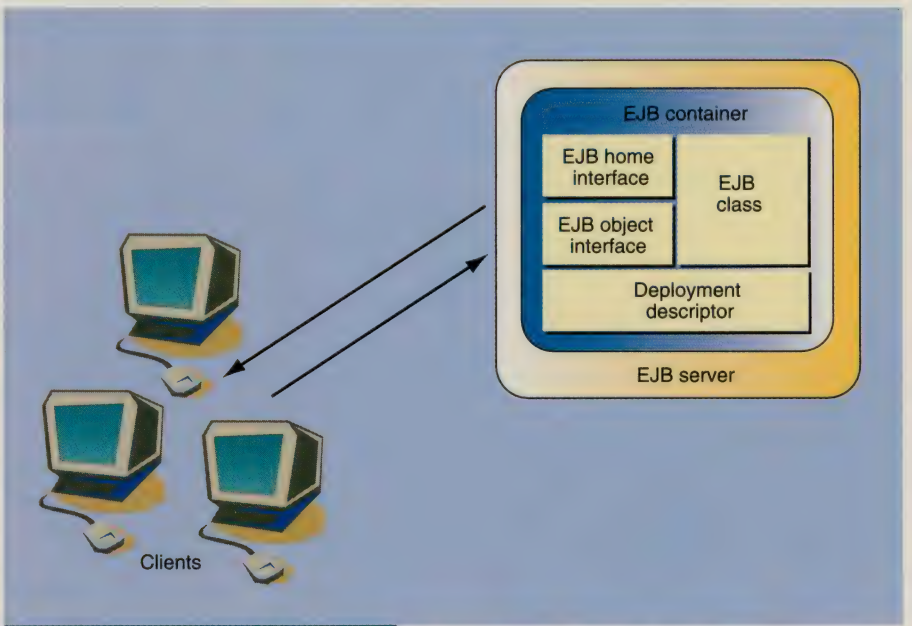


Figure 1 Synchronous EJB Interaction This figure shows an EJB client application and a session/entity EJB bean in a typical interaction.

Listing 1 Prepare the Client This client application enlists a MessageListener with a JMS topic to receive and process messages.

```
import javax.jms.*;

/**
 *  A sample that demonstrates how to obtain a JMS
 *  connection and enlist a message listener. In this case
 *  we are acquiring a connection to a JMS topic.
 */

public class JMSSample {

  public static void main (String args[])
  {
    // create an initial context. The user can specify the
    // JNDI SPI settings for the given JMS vendor using the
    // "-D" command line attributes, e.g., "java -
    // Djava.naming.factory.initial=…"
    InitialContext context = new InitialContext();

    // lookup the topic
    Topic topic (Topic)context.lookup("MyTopic");

    // obtain the topic connection factory that we'll use
    // to create a JMS connection.
    TopicConnectionFactory tcf =
      (TopicConnectionFactory)context.lookup(
      "TopicConnectionFactory");

    // create the JMS connection
    TopicConnection conn = tcf.createTopicConnection();

    // create the JMS session from the connection. In this
    // case we are creating a non-transacted, auto-
    // acknowledging connection.
    TopicSession session = conn.createTopicSession(
      false, Session.AUTO_ACKNOWLEDGE);

    // create the topic subscriber
    TopicSubscriber subscriber =
      session.createSubscriber(topic);

    // enlist our listener
    subscriber.setMessageListener(myListener);

    // indicate to the connection that we are ready to
    // receive messages
    conn.start();
  }
}
```

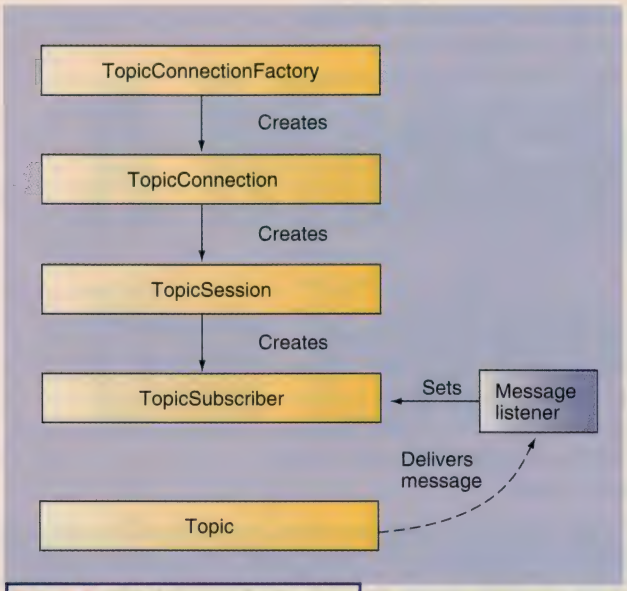


Figure 2 JMS Listener Application To receive messages from a JMS Topic, the application developer needs to follow a prescribed procedure. A message listener can then be enlisted with the resultant TopicSubscriber.

Developing a robust JMS client can be a difficult task. The programmer has to worry about more than just receiving a message. Other areas that need to be addressed include transactions, security, concurrent message processing, object lifecycle, fault tolerance, and scalability—all things that developers have come to expect from an EJB server. Until now, programmers had to combine these technologies by hand.

To integrate EJB 1.1 and JMS, a JMS listener has to be set up in the way we described in Listing 2. The JMS client must reference a stateless session bean that is responsible for processing the JMS message. The JMS message then has to be passed to the EJB. However, JMS messages are not required to be serializable, which means that the message may have had to be converted to a serializable message class or partially deconstructed in the message listener before the message is passed to the remote EJB instance. In addition, the application programmer is responsible for managing transactional associations between the JMS server and the processing EJB, message acknowledgement, and concurrency. These are complex issues and are not necessarily within every developer's domain of expertise.

Although an application developer who completes all these steps has the ability to access the JMS provider and receive messages, obviously a significant amount of coding is involved.

The release of the EJB 2.0 specification resolved these challenges by extending the EJB component model to provide simplicity to bean developers needing support for asynchronous messaging—the new message-driven bean type.

Message-Driven Beans

Message-driven beans are deployed as clients to a particular topic or queue that always plays the role of a message consumer. There is no client view to a message-driven bean, meaning that there are no home and remote interfaces. A message producer writes to a topic or queue and has no knowledge of the fact that a message-driven bean is acting as a message consumer (see Figure 3). This leads to loose coupling between JMS-based systems, allowing for more flexibility in assembling a distributed computing environment.

Message-driven beans have no conversational state. All bean instances are equivalent when they are not processing a message. This is analogous to the state characteristics of stateless session beans; pooling of bean instances is a natural way to manage message-driven bean instances efficiently.

A message-driven bean must be derived, either directly or indirectly, from the interface javax.ejb.MessageDrivenBean. This interface is in turn derived from the javax.jms.MessageListener interface and adds two methods of its own. The onMessage() method is inherited from the javax.jms.MessageListener interface. The sole parameter to the method is a javax.jms.Message, which may be of any valid JMS message type. The method signature does not include a throws clause, so no application exceptions may be thrown during message processing.

As the container receives messages, it first accesses a message-driven bean from a pool of available instances (see Figure 4). The container then associates the role designated in the deployment descriptor to the thread of execution, enabling the propagation of a security context. In addition, the container sets up an association with a transactional context if the deployment descriptor requires one.

Once these administrative tasks are completed, the received message is passed to the onMessage() method of the MDB instance. Once the method completes, the transaction is committed, or rolled back (if a transaction was initiated), and the bean is returned to the pool of available instances.

Another key benefit afforded by message-driven beans is simplicity of deployment

The ejbRemove() method is invoked when a message-driven bean instance will be evicted from any strong references in the container (usually from the instance pool) and made available for finalization and garbage collection. The ejbRemove() method should release any resources held by the bean instance.

The setMessageDrivenContext() method takes one argument—an instance of the

Listing 2 A Good Listener This sample JMS listener retrieves a stock quote from the incoming JMS TextMessage and displays it to standard output.

```
/**
 * This class is an implementation of a JMS
 * MessageListener that processes incoming test
 * messages containing a stock quote.
 */

class MyListener implements MessageListener {

  /**
   *  Pulls a stock quote from the incoming message and
   *  displays it to the standard output stream.
   *
   *  @param message. The incoming message.
   */
  public void onMessage(Message message) {
    // Pull out the quote from the message object.
    // We know that the message producer sends TextMessages
    // and therefore cast the object accordingly.
    try
    {
      String quote = ((TextMessage)message).getText();
      System.out.println("Stock quote: " + quote);
    }
    catch(JMSException e)
    {
      System.out.println(
        "Error processing message: "+message);
    }
  }
}
```

THE PRESSURE TO LAUNCH IS IMMEASURABLE.

WebLoad Resource Manager® can help.

You can test and analyze your site throughout the development process. Allowing teams to communicate faster. Collaborate smarter. And correct problems sooner. So you can deliver your site on time, on spec, and on budget. Because getting it out and getting it right shouldn't be mutually exclusive. Intrigued? Call 888-RadView. Or for a free evaluation, visit www.radview.com.

WebLoad Resource Manager®

RADVIEW™

javax.ejb.MessageDrivenContext class. The MessageDrivenContext class is analogous to the entity and session context classes defined in EJB 1.1. When a bean instance is created, the container passes in a context for the instance to hold. The context class has methods to get environment information and the JTA UserTransaction class used for bean managed transaction demarcation.

In addition, the bean provider must supply an ejbCreate() method with no arguments for the bean to be deployable in an EJB 2.0 server. The bean instance can acquire any resources it needs for processing in the ejbCreate() method. For example, at this point the message-driven bean instance could acquire a database connection that it would close and release if ejbRemove() were called.

Note that the message-driven bean specification has now greatly simplified the creation of JMS message consumers. The programming tasks in Listing 3 for creating and deploying a JMS message consumer have been entrusted to the EJB container; a developer can now simply implement the message-driven bean interface and deploy it to the EJB server to create a message-receiving business component.

An example of a message-driven bean instance implementation is shown in Listing 4. This bean gets a string from a JMS TextMessage and prints the output to standard out. It is an adaptation of the JMS message listener shown in Listing 2.

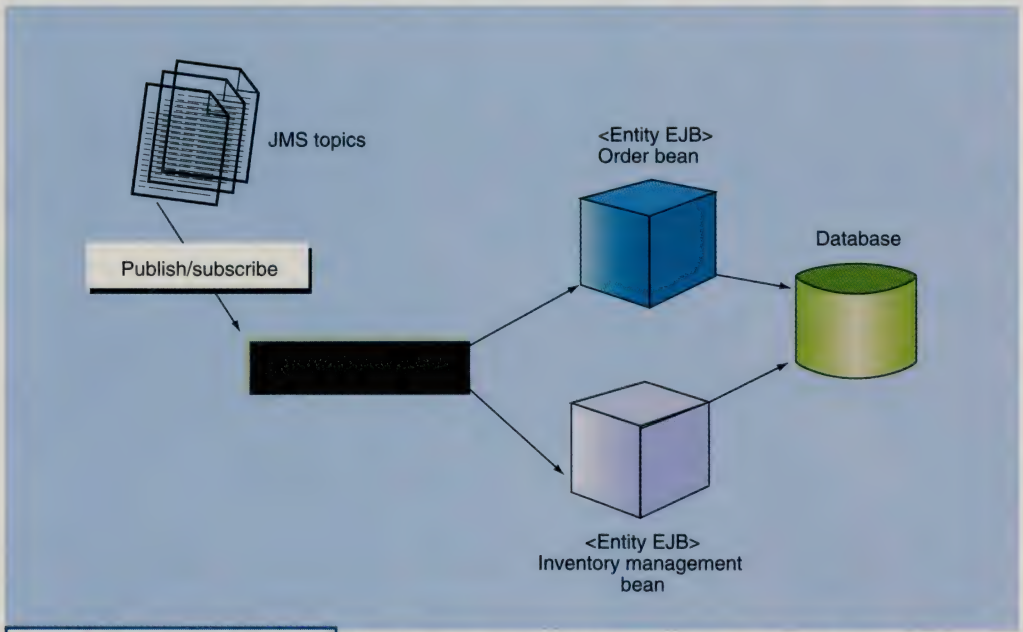


Figure 3 Beans with a Message Message-driven beans are stateless beans that are activated by receiving a message. They may subsequently access other EJBs and resources while processing the contents of the message.

Deployment Descriptor

Message-driven beans specify the runtime behavior controlled by the EJB server information with an XML deployment descriptor. The validating DTD element definition for message-driven beans in the deployment descriptor looks like this:

```
<!ELEMENT message-driven (description?,
  display-name?, small-icon?,
  large-icon?, ejb-name?, ejb-class,
  transaction-type, transaction-scope?,
  jms-message-selector?, jms-acknowledge-
  mode?, message-driven-destination?,
  env-entry*, ejb-ref*, security-
  identity?, resource-ref*, resource-
  env-ref*)>
```

The important thing to notice is that the deployment descriptor contains all the information except the destination name required to deploy a message-driven bean. The destination name is set in an application server's vendor-specific configuration file or as a system property.

In the deployment descriptor, the deployer can specify whether the bean is intended for a topic or a queue and, if intended for a topic,

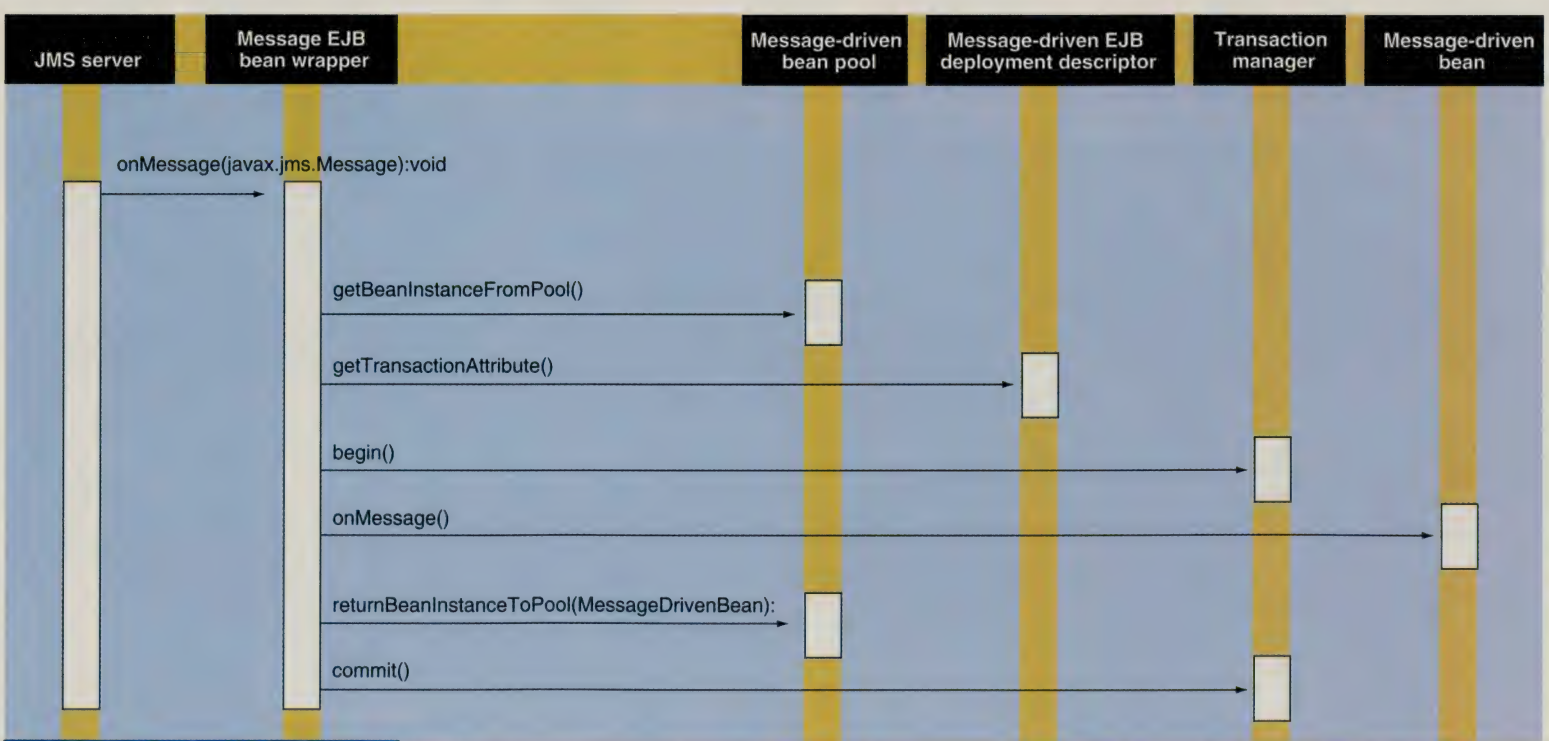


Figure 4 Inner Workings of a Message As this sequence diagram shows, the EJB container manages the bean instances and message delivery.

integrated with System Definition integrated with **Software Development** integrated with Content Management integrated with System Testing integrated with Best Practices integrated with **Software Development** integrated with System Definition

System Testing integrated with Content Management integrated with **Software Development** integrated with Best Practices integrated with Change Management

Developer Controlled-Synchronization of JAVA code and models. A perfect exchange.

Rational Rose® is the #1 JAVA™ development and Visual Modeling tool. IDC names Rational Rose the market leader for the fifth year in a row.* Now you can control the synchronization between code and models. Choose the entire model or select model elements. Turn it on all the time or only when you want. Rational Rose is tailored for easy and efficient developer use. It packs the power to help you build better software, faster. Gain full life-cycle support for JAVA source code including J2SE™ and J2EE™ platforms. Resilient and component-based architecture developed with Rational Rose substantially reduces application rewrites. Put Rational Rose on your team and gain the flexibility, extensibility and compatibility that professional JAVA developers choose first.

Rational®
the e-development company™

JDJ READERS' CHOICE AWARD · JDJ EDITOR'S CHOICE AWARD · 1999 JAVAPro READER'S CHOICE · Java REPORT 2000 WRITERS' CHOICE

Log-on to www.rational.com/javapro for more information and a JAVA white paper and CD.

© 2000 Rational Software Corporation. Rational, the Rational logo, Rational the e-development company, Rational Rose among others are trademarks or registered trademarks of Rational Software Corporation in the United States and/or other countries. J2SE, J2EE, Java and all Java-based trademarks and logos, among others, are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and/or other countries.

*IDC Report 22438: Application Design and Construction Tools Forecast and Analysis, 2000-2004

whether the bean should act as a durable subscriber. Like queues, durable topics guarantee that the listener will receive all messages published to the topic, even if the listener is unavailable for a period of time. Durable topics are important for reliable applications.

The deployment descriptor for our sample message-driven bean (see Listing 5) tells the container that this bean is designed to listen on a nondurable topic. The bean has container-managed transaction demarcation with the NotSupported method transaction attribute.

Another key benefit afforded by message-driven beans is the simplicity of deployment. Typically, an application server vendor will provide a tool to define the deployment descriptor for the bean, generate the EJB jar file, and deploy the bean. Once the bean is deployed, the EJB

Listing 3 Necessary Interface This is the interface a message-driven bean developer must implement. It creates and deploys a JMS message consumer.

```
/**
 * The MessageDrivenBean interface is implemented by every
 * message-driven enterprise Bean class. The container
 * uses the MessageDrivenBean methods to notify the
 * enterprise Bean instances of the instance's life
 * cycle events.
 */

public interface javax.ejb.MessageDrivenBean extends
  javax.jms.MessageListener
{
  /**
   * Pass a message to the listener.
   *
   * @param message The message passed to the listener.
   */
  public void onMessage(javax.jms.Message message);

  /**
   * A container invokes this method before it ends the
   * life of the message-driven object. This happens when
   * a container decides to terminate the message-driven
   * object.
   *
   * This method is called with no transaction context.
   */
  public void ejbRemove();

  /**
   * Set the associated message-driven context. The
   * container calls this method after the instance
   * creation.
   *
   * The enterprise Bean instance should store
   * the reference to the context object in an
   * instance variable.
   *
   * This method is called with no transaction context.
   */

  public void
    setMessageDrivenContext(
    javax.ejb.MessageDrivenContext context);
}
```

Listing 4 Not-So-Hidden Message This bean extracts the text from a message and prints the contents to standard output.

```
/**
 * This class is an implementation of a MessageDrivenBean.
 * It derived from the JMS MessageListener shown in
 * Listing 2.
 */

public class MyListenerMDB implements MessageDrivenBean
{
 /**
   * A no-arg constructor is required so that the EJB
   * container can use the Class.newInstance() method to
   * create bean instances.
   */

  public MyListenerMDB()
  {
  }

 /**
   * The method receives the message instances and
   * performs the message processing.
   *
   * @param message   The message object.
   */
  public void onMessage(Message message)
  {
    // The onMessage implementation remains unchangesd:
    // Pull out the quote from the message object.
    // We know that StockQuoteProducer sends TextMessages
    // and cast the object accordingly.
    try
    {
      String quote = ((TextMessage)message).getText();
      System.out.println("Stock quote: " + quote);
    }
    catch(JMSException e)
    {
      System.out.println(
        "Unable to process message: " + message);
    }
  }

 /**
   * This method is invoked when the message-driven
   * bean instance is evicted from the container.
   */
  public void ejbRemove() throws javax.ejb.EJBException
  {
    System.out.println(
      "StockListenerMDB: ejbRemove called.");
  }

 /**
   * Sets the MessgageDrivenContext instance. This method
   * is Invoked by the container during the instantiation
   * of the Bean instance. The message-driven context
   * allows the bean Developer to access the facilities
   * of the EJB container (similar to the EJB context of
   * the entity and session beans).
   *
   * @param ctx   The message-driven context.
   */
  public void setMessageDrivenContext(
    MessageDrivenContext ctx) throws javax.ejb.EJBException
  {
    System.out.println(
      "StockListenerMDB: setMessageDrivenContext called.");
  }

 /**
   * ejbCreate with no args required by spec, though not
   * enforced by interface
   */
  public void ejbCreate()
  {
    System.out.println(
      "StockListenerMDB: ejbCreate called.");
  }
}
```

Paying too much for J2EE™ technology is a little dinghy

The next wave is Unify eWave

Affordable. Scalable. Production-ready.

Unify eWave

Get your FREE developer downloads TODAY:

www.UnifyeWave.com

Java and all Java-related marks are trademarks or registered trademarks of Sun Microsystems in the US and other countries

Listing 5 Application for Deployment The deployment descriptor tells the container how to manage the runtime behavior of the message-driven bean.

```
<ejb-jar>
  <enterprise-beans>
    <message-driven>
      <ejb-name>MessageListenerMDB</ejb-name>
      <ejb-class>messageListenerMDB</ejb-class>
      <transaction-type>Container</transaction-type>
      <transaction-scope>Local</transaction-scope>
      <jms-acknowledge-mode>auto-acknowledge</jms-
        acknowledge-mode>
      <message-driven-destination>
        <jms-destination-type>javax.jms.Topic</jms-
          destination-type>
        <jms-subscription-durability>nondurable</jms-
          subscription-durability>
      </message-driven-destination>
    </message-driven>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>MessageListenerMDB</ejb-name>
        <method-name>onMessage</method-name>
        <method-params>
          <method-param>javax.jms.Message</method-param>
        </method-params>
      </method>
      <trans-attribute>NotSupported</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

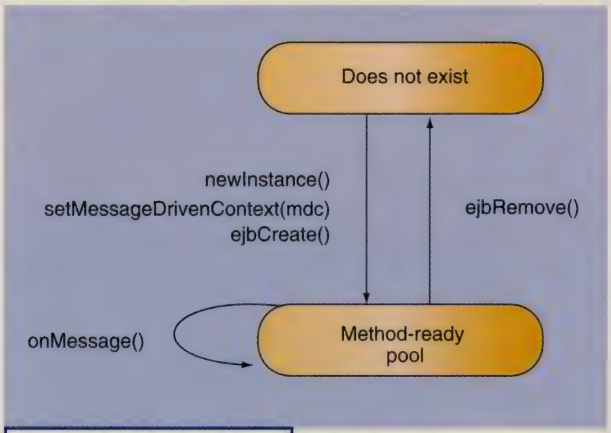


Figure 5 The Life of a Bean This is a state transition diagram showing the lifecycle of a message-driven bean.

server handles the registration of the EJB container classes and the start of the JMS connection. Since JMS defines a standard, a JNDI-based mechanism for acquiring references to JMS topics, queues, and connections, a high-end application server can work with any JMS vendor's implementation seamlessly out of the box. This allows architects to make best-of-breed choices in application and JMS servers when developing a business solution.

Message-Driven Bean Container Services

The EJB container provides the following services based on the declarative deployment descriptor entries in the EJB jar file. These are low-level EJB support services that were not directly available to JMS developers in EJB 1.1.

- **Lifecycle management:** The lifecycle of a message-driven bean corresponds to the lifespan of the EJB server in which it is deployed. Since message-driven beans are stateless, bean instances are typically pooled by the EJB server and retrieved by the container when a message becomes available on the topic or queue for which it is acting as a message consumer (see Figure 5). The container creates a bean instance by invoking the newInstance() method of the bean instance class object. After the instance is created, the container creates an instance of javax.ejb.MessageDrivenContext and passes it to the bean instance via the setMessageDrivenContext() method. The bean instance is then placed in the pool and is available to process messages.
- **Exception handling:** Message-driven beans may not throw application exceptions while processing messages. This means that the only exceptions that may be thrown by a message-driven bean are runtime exceptions indicating a system-level error. The container will handle this exception by removing the bean instance and rolling back any transaction started by the bean instance or by the container.
- **Threading and concurrency:** A message-driven bean instance is assumed to execute in a single thread of control, which greatly simplifies the bean developer's task. The EJB server will guarantee this behavior. In addition, the EJB server may provide a mode of operation that allows multiple messages to be handled concurrently by separate bean instances. This deployment option uses "expert level" classes defined in the JMS specification. The JMS provider is not required to provide implementations for these classes, so the EJB server may not be able to take advantage of them with every JMS implementation. Using these classes involves a tradeoff between performance and serialization of messages delivered to the server.
- **Transactions:** As with entity and session beans, message-driven beans can have container- or bean-managed transactions. A transaction attribute may be set for the message-driven bean's onMessage() method. Because there are no client transactions, only a subset of the transaction attributes allowed for entity and session beans are relevant for message-driven beans. The container will support the Required and NotSupported transaction attributes for message-driven beans with container-managed transactions.

A message-driven bean with bean-managed transactions can use the JTA UserTransaction object acquired from the bean's context to demarcate the transaction boundaries during message processing. A message-driven bean with bean-managed transactions must end the transaction with either a commit or rollback before returning from the onMessage() method.

The implementation of an onMessage() method with container-managed transaction demarcation and the NotSupported transaction attribute is very simple. The container will create no transactions and the bean developer is prohibited from accessing the UserTransaction object.

Message-driven EJBs could become the de facto component model for handling *any* kind of message, not just messages delivered by a JMS server

The more interesting case is when the onMessage() method is assigned the transaction Required attribute. In this case, the container will create a global transaction that will enlist any referenced resources and be propagated to any other EJBs that are used during message processing. This is the only scenario in which the interface to the JMS topic or queue is involved in the EJB transaction.

Make Java on the Server More Potent!

AppStream for Java™ Technologies
enables breakthrough performance of Java-based network computing.

www.appstream.com

AppStream™

1-866-AppStream *(toll free)*

©2000 AppStream. AppStream is a trademark of AppStream. All other marks are trademarks of their respective owners.

For beans with the Required attribute, the JMS server will be enlisted as an XAResource. If the JMS server provider does not support XA transactions, the commit or rollback on the JMS session will typically be synchronized with the result of the container's global transaction. In this case, the container will not be able to involve the JMS server in the two-phase commit process used in managing distributed transactions. A rollback on the JMS session will alert the JMS server that the message should be redelivered.

- **Message acknowledgement:** The container always handles message acknowledgement for message-driven beans; it is illegal for the bean to use the client message acknowledgement methods defined in the JMS specification. The message acknowledgement can be set to DUPS_OK_ACKNOWLEDGE or AUTO_ACKNOWLEDGE. The former allows the delivery of duplicate messages after a failure, while the latter provides a strict guarantee that the message will be delivered only once. As noted, a bean with the Required transaction attribute assigned to the onMessage() method will involve the message delivery in a transaction.
- **Security:** Because the message-driven bean has no client, no security principal is propagated to the container on receipt of a message. The EJB 2.0 specification provides facilities for a bean method to execute in a declaratively specified role. As a result, the message-driven bean can be configured to execute in a security context that can be propagated to other EJBs used during message processing. This allows method-level security to be maintained for message-driven beans.

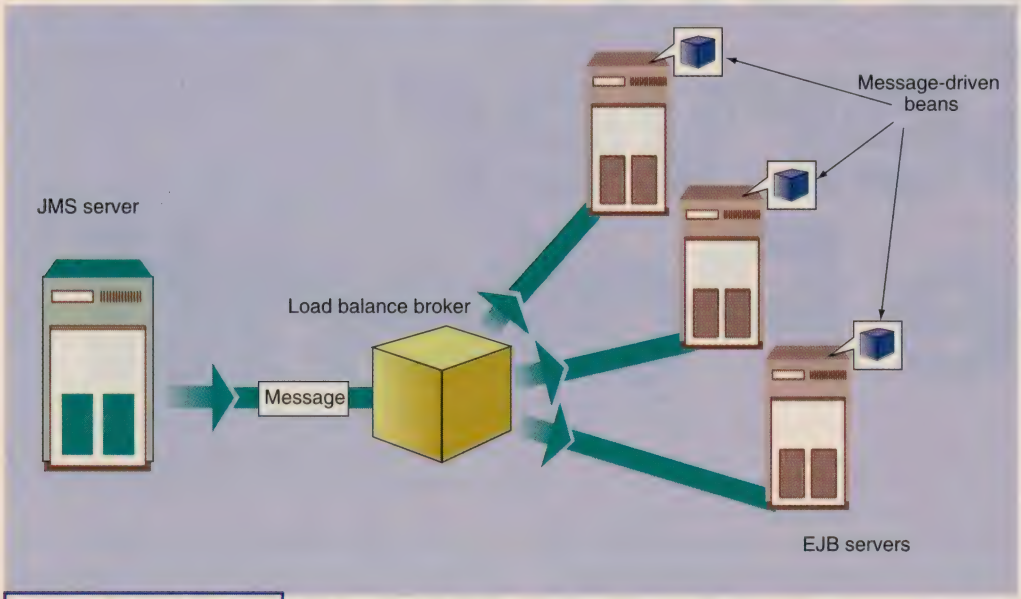


Figure 6 Balance the Load The incoming JMS messages can be distributed to awaiting EJB server instances. These instances host the message-driven beans that handle the processing of the message contents.

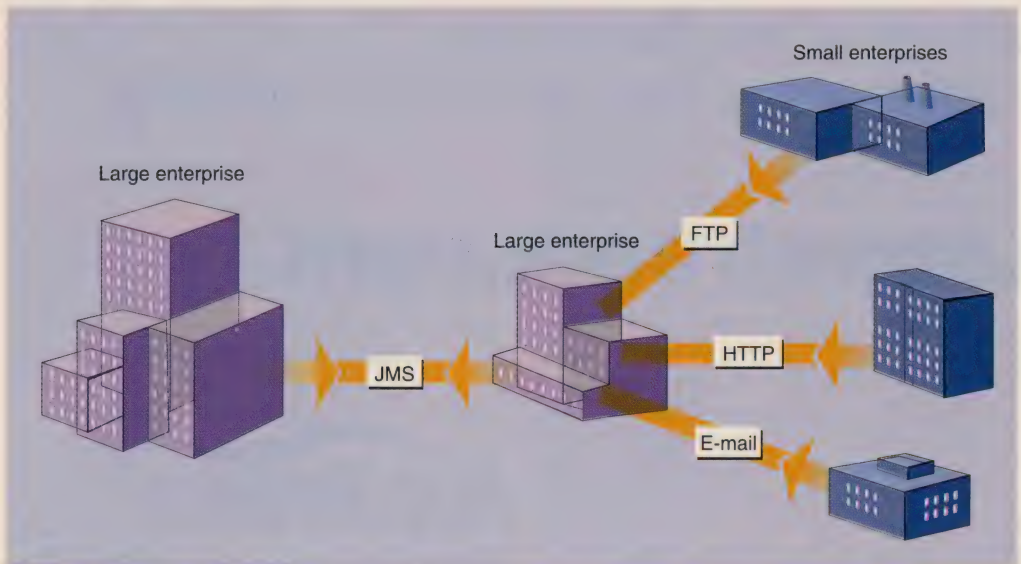


Figure 7 Business as Usual In a typical B2B scenario, many businesses may play the role of a "spoke" around a large enterprise acting as a "hub." Some of the spokes can interact directly with the hub via JMS. Other spokes may interact via e-mail or HTTP.

Application Server Framework

When deployed within the framework of an application server, the functionality provided by message-driven beans expands exponentially. A high-end application server provides scalability, load balancing, dynamic application start-up, dynamic configuration of additional EJB server instances and fault-tolerance, which are essential for enterprise-class applications. (Note: The requirements for these features are not defined by the EJB 2.0 specification, and the degree to which they are provided for EJBs in general will vary with the application server implementation.)

- **Scalability:** A key element of an enterprise application server is its ability to provide a framework to accommodate increasing processing load. As the number of messages being processed by message-driven beans increases, the application server's EJB server must accommodate the increase without a significant degradation in processing time. Scalability can be afforded to an application by offering multiple application server instances that work in concert to process the incoming messages.
- **Load balancing:** As requests start flowing into the application server, a load-balancing element can distribute the load evenly among these instances (see Figure 6). For example, each instance of the application server could contain an instance of an EJB server. Using the application server's load-balancing facilities, incoming messages can be distributed across all EJB server instances.
- **Dynamic application start-up:** An application server should be able to start additional, preconfigured instances as load increases.
- **Dynamic configuration of additional EJB server instances:** Even if an application is configured with an appropriate number of instances to handle the typical application load, a situation may arise where a spike in message flow may cause the system processing time to degrade. To handle that situation, an application server must have the management facilities for deploying and configuring additional application server instances. For example, the application server could allow the system administrator to dynamically define additional EJB server instances and deploy EJBs to those additional instances. The load-balancing element will start utilizing these supplementary instances for message processing.
- **Fault tolerance:** An enterprise application server must also be able to accommodate aberrant situations, such as network or hardware failures, without significant degradation in overall system performance and without the loss of messages. One way in which an applica-

User assistance for Java apps really can be easy!

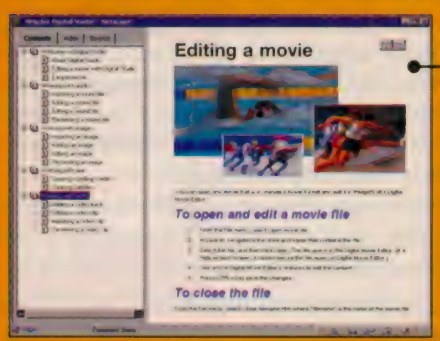
NEW RELEASE!

RoboHELP® Office 9.0 is your **Help** and **documentation** solution.

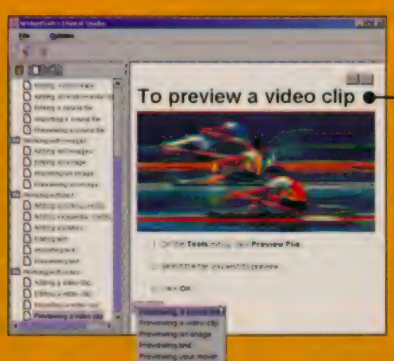
Just point and click to develop the Help content, structure and design – no coding. A few clicks let you generate the finished Help system in the format you need in seconds. It's that simple.

So whatever format you choose to deploy, RoboHELP Office gives you the flexibility you need to decide. While making it fast and easy for you to develop and deliver high quality Help and documentation to your end users.

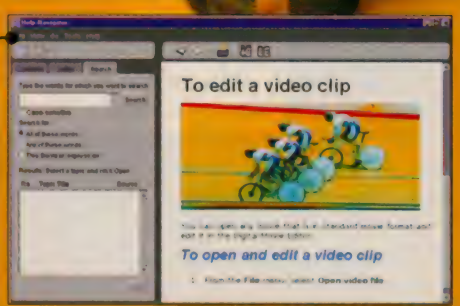
A few mouse clicks let you generate all of this from the same source project!



WebHelp
eHelp's versatile cross-platform, cross-browser Help format. Deploy it anywhere. Can be web-based and server-based.



JavaHelp
Developed by Sun Microsystems as their user assistance standard. A 100% pure Java solution.



Oracle Help for Java
Not just for Oracle developers. A 100% pure Java solution developed by Oracle with robust functionality. Not Oracle dependent.

Printed Documentation
RoboHELP Office 9.0 lets you generate printed documentation directly from your Help project.

eHelp Corporation

Try RoboHELP Office FREE!

Get a free 15-day license of RoboHELP Office 9.0

Visit **www.ehelp.com/try9free** or call **800-563-0590**

Listing 6 Next order of Business This business order is provided in XML. XML provides a flexible and open medium for businesses to communicate.

```
<purchase_order>
  <client_number>101</client_number>
  <date>August 18, 2000</date>
  <billing_address>
    <name>Joe Programmer</name>
    <street>1000 Briggs Road</street>
    <city>Mt. Laurel</city>
    <state>NJ</state>
    <zip>08054</zip>
  </billing_address>
  <shipping_address>
    <name>Joe Programmer</name>
    <street>1000 Briggs Road</street>
    <city>Mt. Laurel</city>
    <state>NJ</state>
    <zip>08054</zip>
  </shipping_address>
  <items>
    <item>
      <quantity>1</quantity>
      <product_number>324</product_number>
      <description>Java IDE</description>
      <unitcost>99.95</unitcost>
    </item>
    <item>
      <quantity>1</quantity>
      <product_number>532</product_number>
      <description>Java VM Heap Profiler</description>
      <unitcost>56.95</unitcost>
    </item>
  </items>
</purchase_order>
```

tion server can provide this facility is through the deployment of multiple application server instances that are able to share state information. Incoming messages are distributed to the instances via an intelligent load-balancing element. If one or more instances becomes inaccessible, the load balancer can simply redistribute the messages among the available instances. In addition, the load-balance broker can work with an administrative component to restart instances should they fail for some reason.

Beyond JMS

The message-driven bean portion of the EJB specification is one of the most exciting and powerful additions to enterprise computing in recent years. It allows messages received via a JMS implementation to be handled by a simple and powerful component model. However, message-driven EJBs have the potential to become the de facto component model for handling *any* kind of message, not just messages delivered by a JMS server.

JMS messages are designed to be interoperable across JMS implementations. This means that a JMS message may have a proprietary implementation under the covers, but the contents can be transparently converted to another proprietary JMS implementation without affecting the message consumer in any way. Since this conversion can be done from one JMS implementation to another prior to message delivery to the client, any arbitrary message format can be converted to a JMS message and delivered to a message-driven bean.

In other words, a message-driven bean can process messages sent by e-mail, HTTP, FTP, or any other protocol, provided the application server has the facility to convert these protocols into a JMS message. This opens the door to a standard, simple, portable component model that can process any message delivered by any protocol. If the messages are defined in an open, extensible language like XML, unprecedented interoperability can be achieved in loosely coupled systems in a model that everyone understands. A simple use case for this protocol-independent message processing capability illustrates the power of this technology.

A B2B Use Case

A typical business-to-business scenario involves a hub-and-spoke topology (see Figure 7), where multiple loosely coupled businesses (the "spokes") interact with large enterprises (the "hubs"). In some cases, it may be appropriate for some of the hubs to interact regularly with the spoke via JMS applications when performing interactions such as submitting large orders. For small businesses, this pure JMS approach may be cost prohibitive. It might be better to simply submit small orders via HTTP from a Web browser or send an e-mail consisting of an order. The enterprise hub should be able to work with an array of protocols to open its business-to-business system to any customer.

The message-driven bean portion of the EJB specification is one of the most exciting and powerful additions to enterprise computing in recent years

Moreover, it does not make sense for the hub enterprise to have to write separate message handlers for the same order form just because one order arrives via e-mail and another via JMS. Using message-driven beans, this is not necessary—any protocol can be used to send an order described in XML, like the example in Listing 6. The message-driven bean need only be concerned with processing the XML from within the JMS TextMessage that the application server delivers, independent of whether it was originally delivered via a JMS server or as an SMTP message from a mail server.

The EJB 2.0 specification provides a new component model—the message-driven bean—for processing asynchronous messages. Message-driven beans are simple to develop and take advantage of the same container services as other EJB types. This new component model allows integration with legacy message-oriented middleware as well as with the new generation of all-Java JMS implementations that have emerged over the last year. And, while originally intended to act as an integration point for JMS, message-driven beans have the potential to become the industry standard for processing any asynchronous message, particularly XML-based messages.

Editor's note: This article is based on the initial public draft of the EJB 2.0 spec. Since the article was written, the final public draft has become available. In a future article, the authors will discuss the changes introduced in this final public draft. ■

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101JM Download the code for this article.

JP0101JM_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Become a Subscriber NOW!

JAVA Pro™

Yes! ☑ Send me one year (13 issues) of *JAVA Pro* for just $29.95

Name: _____

Address: _____

City/ State/ Zip: _____

Country: _____ E-mail: _____

❑ Bill Me ❑ Payment Enclosed Charge my: ❑ Visa ❑ MC ❑ AMEX

Card Number: _____ Exp. _____

Signature: _____

One year is 12 issues plus 1 special issue. Annual cover price is $77.35. Make checks payable to JavaPro. Please allow 4-6 weeks for delivery of first issue. International subscription must be payable in U.S. Dollars plus postage. Canada/Mexico add $18/year; all other countries add $44/year for air mail. Canadian GST included. Java is a trademark of Sun Microsystems.

11/99

4184

Become a Subscriber NOW!

JAVA Pro™

Yes! ☑ Send me one year (13 issues) of *JAVA Pro* for just $29.95

Name: _____

Address: _____

City/ State/ Zip: _____

Country: _____ E-mail: _____

❑ Bill Me ❑ Payment Enclosed Charge my: ❑ Visa ❑ MC ❑ AMEX

Card Number: _____ Exp. _____

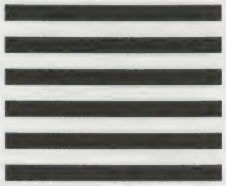
Signature: _____

One year is 12 issues plus 1 special issue. Annual cover price is $77.35. Make checks payable to JavaPro. Please allow 4-6 weeks for delivery of first issue. International subscription must be payable in U.S. Dollars plus postage. Canada/Mexico add $18/year; all other countries add $44/year for air mail. Canadian GST included. Java is a trademark of Sun Microsystems.

11/99

4184

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

POSTAGE WILL BE PAID BY ADDRESSEE

JAVA™ Pro
PO BOX 54584
BOULDER CO 80323-4584

Subscribe online at http://www.java-pro.com

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

POSTAGE WILL BE PAID BY ADDRESSEE

JAVA™ Pro
PO BOX 54584
BOULDER CO 80323-4584

Subscribe online at http://www.java-pro.com

The Hottest Magazine for the Hottest Developers in the Enterprise

If you're a professional developer programming in the hottest language in the fastest growing development market in the universe, then you need the hands-on, code-intensive, product-oriented information that comes only from *Java Pro* magazine. Harness Java's unique capabilities in YOUR enterprise with no-holds-barred editorial content.

Don't miss this special offer! Get 13 information-packed issues—a full year of Java Pro *—for only $27.95. That's a savings of more than 61% off the cover price. Call today!*

Java™ Pro: get beyond the hype

- Practical techniques to amplify Java performance
- Independent, objective product reviews to help you find only the best products for your needs
- Complete source code examples using JavaBeans, servlets and more
- Tips and tricks for Java power programmers

***Java Pro* is the professional developers independent source for building serious Java applications**

Order online today
www.java-pro.com

Subscribe by phone
1-800-848-5523

FTP FAWCETTE TECHNICAL PUBLICATIONS
209 Hamilton Ave.
Palo Alto, CA 94301

Improve Java Performance

Good coding principles, including better use of classes and improved thread pooling, can bring substantial performance gains

by Tarak Modi

Java's ability to let programmers develop complex applications relatively easily has no doubt contributed to its widespread popularity. However, this ease of use is a double-edged sword: Java critics often complain that "Java won't perform." A well-written program in Java may not perform as well as the same well-written program in C++, but most performance problems in Java programs are due not to Java, but to the program itself. Good design principles can go a long way in improving performance. For example, adding a thread pool to a Java application can not only improve performance, but make the application more robust and platform independent as well (see "Why Thread Pools are Important in Java," *Java Pro*, October 2000). However, good coding practices—including better use of the java.lang.String and java.util.Vector classes—must also be mastered, and can significantly bump up performance as well. In other words, a well-designed program will not, and cannot, compensate for poor programming practices. Let's see how these ideas work.

Coding for Performance

Probably one of the most used and abused classes in Java is java.lang.String. This is also the class that leads to the most inefficiencies in program code. Consider this:

```
String s1 = "Testing String";
String s2 = "Concatenation Performance";
String s3 = s1 + " " + s2;
```

Almost every Java programmer knows that this is inefficient. So what should we do? Maybe something like this:

```
StringBuffer s = new StringBuffer();
s.append("Testing String");
s.append(" ");
s.append("Concatenation Performance");
String s3 = s.toString();
```

This code will perform much better than the first snippet, right? Wrong. This is exactly what the compiler does for you in the first code snippet. If using a StringBuffer does not make the code any more efficient than using the separate String objects, why does every Java book condemn the first code snippet and favor the second?

Let's look at the default StringBuffer constructor used in the second code snippet (and by the compiler in the first snippet). This is what it looks like:

```
public StringBuffer() {
  this(16);
}
```

The default constructor assumes a capacity of 16 characters. Now let's look at the append() method of the StringBuffer class.

```
public synchronized StringBuffer append(
  String str) {
  if (str == null) {
      str = String.valueOf(str);
  }
  int len = str.length();
  int newcount = count + len;
  if (newcount > value.length)
    expandCapacity(newcount);
  str.getChars(0, len, value, count);
  count = newcount;
  return this;
  }
```

Tarak Modi is a lead systems architect at Online Insight. A certified Java programmer, he has several years of experience working with Java, C++, and technologies such as Enterprise JavaBeans, CORBA, and DCOM. He holds a bachelor's degree in electrical engineering, a master's degree in computer engineering, and an MBA with a concentration in IS. He can be reached at Tarak.Modi@onlineinsight.com.

The append() method first calculates the total length of the final appended string; if this is greater than what the StringBuffer can store, it calls the private expandCapacity() method. This method doubles the capacity of the StringBuffer each time it is called and copies the existing character array in the new memory.

In the second code snippet (and in the compiler-generated code in the first snippet), the StringBuffer capacity would have to be expanded two times, resulting in two expensive copies, because the length of the final String "Testing String Concatenation Performance" is 40 characters. Thus one way we can beat the compiler is by allocating a StringBuffer with an initial size greater than or equal to 40 characters, like this:

```
StringBuffer s = new StringBuffer(45);
s.append("Testing String");
s.append(" ");
s.append("Concatenation Performance");
String s3 = s.toString();
```

Consider this:

```
String s = "";
int sum = 0;
for(int I=1; I<10; I++) {
  sum += I;
  s  = s + "+" I ;
}
s = s + "=" + sum;
```

Think about why that snippet is less efficient than this:

```
StringBuffer sb = new StringBuffer()
int sum = 0;
for(int I=1; I<10; I++){
  sum + = I;
  sb.append(I).append("+");
}
String s = sb.append("=").append(sum).toString();
```

The reason is that each s = s + "+" I execution results in the creation and destruction of one StringBuffer and one String object. This is pure waste, which we avoid in the second case.

Let's look at another popular Java class—java.util.Vector. In the simplest terms, a Vector is an array of java.lang.Object instances. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate the addition and removal of items after the Vector has been created. Consider the case of adding an element to a Vector as shown here:

```
Object obj = new Object();
Vector v = new Vector(100000);
for(int I=0; I<100000; I++) {  v.add(0,obj);
}
```

Unless there is an extremely compelling reason to add the new element at the beginning of the Vector each time, this code is a performance hog. The initial capacity of the Vector is set to 10 elements in the default constructor, and the capacity is doubled each time an element is added beyond capacity. As with the StringBuffer class, each time the capacity is expanded, all the existing elements are copied into the new memory. This code snippet would be several orders of magnitude faster than the first example:

```
Object obj = new Object();
Vector v = new Vector(100000);
for(int I=0; I<100000; I++) {
  v.add(obj);
}
```

The same logic applies with the remove() methods of the Vector class, as well. Removing an element at any index other than the last index results in a shifting of all elements beyond the removed element, because a Vector cannot have any "gaps" between its contained elements. This means that removing the last element from a Vector is many times "cheaper" than removing the first element from it.

Let's say you wanted to remove all elements from our Vector example. You could try this:

```
for(int I=0; I<100000; I++){
  v.remove(0);
}
```

which would be several orders of magnitude slower than this:

```
for(int I=0; I<100000; I++){
  v.remove(v.size()-1);
}
```

The best way to remove all elements from v is:

```
v.removeAllElements();
```

Assume that the Vector v contains a String "Hello." Consider this code, which is used to remove this String from the Vector:

```
String s = "Hello";
int i = v.indexOf(s);
if(I != -1)
  v.remove(s);
```

This code looks harmless, but once again it slows performance. Here the indexOf() method does a sequential search in v and finds the index of the String "Hello." Then the call remove(s) does the same sequential search again. A better version is:

```
String s = "Hello";
int i = v.indexOf(s);
if(I != -1)
  v.remove(i);
```

This time we've eliminated the second sequential search by giving the remove method the exact index. An even better version is:

```
String s = "Hello";
v.remove(s);
```

Inefficient coding always degrades performance, but simple remedies can greatly improve results

Finally, let's look at one more code snippet about the Vector class:

```
for(int I=0; I<v.size(); I++) {
  System.out.println(
    v.get(I).getClass().toString());
}
```

If v contains 100,000 elements, this code snippet will call v.size() 100,000 times. Although the size method is fairly lightweight, it is still a function call for which the JVM has to set up and tear down a stack at the very least. Here, the code inside the for loop does not modify the size of the Vector v in any way. So a better way to write this code is:

```
int size = v.size();
for(int I=0; I<size; I++) {
  System.out.println(
    v.get(I).getClass().toString());
}
```

A simple change, but still a performance winner. After all, every CPU cycle counts.

To prove that these suggestions would improve performance, I decided to apply some of them to the thread pool implementation discussed in "Why Thread Pools are Important in Java" and compare the performance of the thread pool before and after the performance improvement changes are made. I made four performance improvements to the thread pool implementation. I did not make any changes to the ThreadPool interface, so existing clients will continue to be unaffected (except that their performance may improve, as well).

Performance Improvement #1

Use java.lang.ArrayList instead of java.lang.Vector class. Synchronization is key to preventing common hazards—especially race conditions—in multithreaded programming. However, too much synchronization can become a performance bottleneck, which is exactly the situation in the current implementation of the thread pool. The list of all the threads in the pool is maintained in a Vector. Every method in the Vector class is synchronized, because Vector is designed to be a thread-safe class. However, we do not need this extra level of synchronization, because every publicly accessible method on the thread pool is synchronized and every access to the Vector objects—such as from the PoolThread instances—are made from synchronized blocks. The JDK, however, gives us an option. The ArrayList class is essentially an unsynchronized Vector, which is exactly what we want.

Performance Improvement #2

Remove jobs from the pending jobs list in reverse order. In the current thread pool implementation, pending jobs are removed from the beginning of the pendingJobs list, or _pendingJobs.remove(0). Since this version slows performance, I will remove pending jobs from the end of the list, or _pendingJobs.remove(_pendingjobs.size()-1). If new jobs are added at a faster rate than jobs are completed, the earlier jobs will not get done. That is a conscious tradeoff in favor of performance. Alternatively, I could have used a user-configured, fixed-capacity circular queue instead of a list to store the pending jobs. In this way, jobs could be removed from the beginning of the queue and we could still avoid the possible problem of earlier jobs not getting done. If the queue ever got full, the thread pool would return an overflow error to the caller. Since there is never any shifting of elements in this queue, and hence expensive memory copies, we're OK as far as performance is concerned.

Performance Improvement #3

Break the one list for all available threads into two lists. The thread pool now maintains one list of all threads. When a new job is added to the pool, the thread pool takes these steps:

1. Finds an idle thread using the private method findFirstIdleThread().
2. If there is no idle thread, checks to see if the pool is allowed to create a new thread.
3. If the pool is allowed to create a new thread, creates one and adds it to the list of available threads. Otherwise just returns.
4. If step 1 found an idle thread, gets that thread, changes its status to busy, and notifies it of the new task.

There are several problems with this structure. Here's the implementation of findFirstIdleThread():

```
private int findFirstIdleThread() {
  int size = _availableThreads.size();
  for( int i=0; i<size; i++ ) {
    if( ((ThreadElement)
      _availableThreads.get(i)).
      _idle )return(i);
  }
  return(-1);
}
```

The code has to loop sequentially through all the elements in the list, and, in the worst case scenario, still return –1, indicating that no idle thread is available. This is an expensive operation. Moreover, step 4 is implemented inefficiently: Instead of getting the element at the given index once and storing it in a temporary object, the code gets the same element three times.

Let's break the one list of all threads into separate lists. One list maintains all the busy threads, _busyThreads, and the other list maintains all the idle threads, _idleThreads. Both these lists will be instances of ArrayList. By breaking the list of all threads into two lists, the need for the expensive findFirstIdleThread() method is eliminated. Instead, to check the existence of an idle thread, do this:

```
if( _idleThreads.size() == 0 ){
  // No idle thread
}
else{
  // At least one idle thread
}
```

Step 4 has also been cleaned up. Also, the last thread from the list of idle threads is removed and added to the end of busy threads list.

Let's look at the implementation of the getStats() method of the thread pool. This method is positively affected in two ways as a result of the performance improvements in this phase. First, we no longer need the private findNumIdleThreads() method, since this method call can be replaced by _idleThreads.size(). Even a glance at the code for the findNumIdleThreads() method is enough to give anybody performance jitters. Calling _idleThreads.size() is much more efficient than the findNumIdleThreads() method. Second, since the thread pool no longer maintains a list of all available threads in the pool (such as _availableThreads), you can no longer make the call _availableThreads.size(). So I added a private method totalThreads(), defined like this:

```
private int totalThreads() {
  return(_busyThreads.size() +
    _idleThreads.size());
}
```

This method will be used in place of _availableThreads.size() calls and is negligibly more expensive than calling _availableThreads.size().

Thus, as a result of the changes made in this phase, the performance of the getStats() method has improved. And, as a result of not having the _availableThreads list anymore, we will have to change the private findMe() and removeMe() methods to compile the code. But wait until the next performance improvement phase, where we can get rid of these methods altogether.

Resources

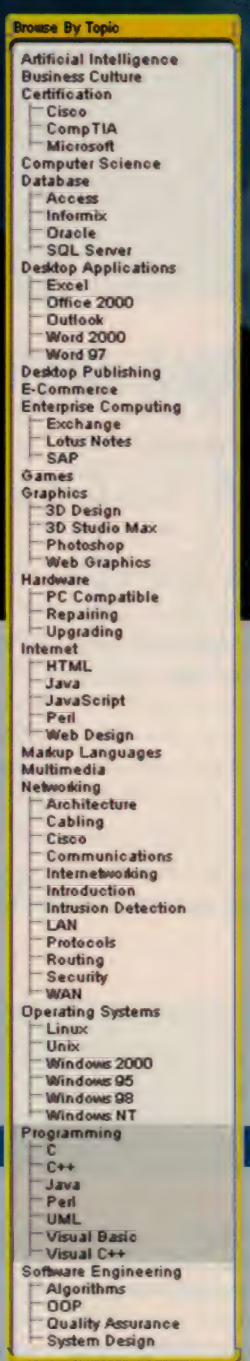
The JDK source code that comes with Sun's JDK in src.jar

Java Platform Performance: Strategies and Tactics by Steve Wilson, Jeff Kesselman (Addison Wesley Longman, 2000)

Java in Practice: Design Styles and Idioms for Effective Java by Nigel Warren, Philip Bishop (Addison Wesley Longman, 1999)

Java 2 Performance and Idiom Guide by Craig Larman, Rhett Guthrie (Prentice Hall, 1999)

The spec requires java applets to communicate to an Oracle database. So what's the best way to access the database using applets? A quick search of InformIT.com gave me the answer. First I needed to choose the right JDBC driver. InformIT.com gave me a quick-to-read table that showed the compatibilities between the different Oracle database versions and the JDBC drivers. Also, I saw, that because I was writing an applet, I had to use the JDBC thin driver because the JDBC OCI-based driver classes can't be downloaded to a Web browser. Now everything works. www.informit.com



©2000-2001 InformIT— A Pearson Company All rights reserved.

InformIT

eCONTENT
EXPERT ACCESS
ONLINE TRAINING

Number of jobs	Original pool (approx. ms)	Improved pool (approx. ms)
50,000	3886	381
200,000	12939	1612

Table 1 Comparing Pools To quantify results, I ran a program called TestPerformance() on my laptop. This program compares the performance between the original pool and the improved pool. These are my averaged results.

Performance Improvement #4

Get rid of the expensive findMe() and removeMe() methods. Why are these methods expensive? Let's examine the findMe() method.

```
private int findMe() {
  int size = _availableThreads.size();
  for( int i=0; i<size; i++ ) {
    if( ((ThreadElement)
      _availableThreads.get(i)).
      _thread == Thread.currentThread() )
    return(i);
  }
  return(-1);
}
```

This method is always called when a thread in the pool is in the busy state and we want to mark it as being idle. Yet the code goes through all the threads available in the pool, which may include many idle threads. In the worst case, the thread we are looking for is the last thread in this list. With the lists broken up into a list of busy threads and a list of idle threads, under most circumstances this search will be faster. Furthermore, with the list broken up, we do not even need this method. We can replace the original code in the PoolThread class that uses the findMe() method with this:

```
if( _pendingJobs.size() == 0 ) {
  if( _debug )
    System.out.println("Idle Thread...");
  ThreadElement elem =
    _busyThreads.remove();
  if( elem == null )
    return;
  elem._idle = true;
  _idleThreads.add(elem);
  break;
}
```

This code simply removes the thread from the list of busy threads, marks it as being idle, and adds it to the list of idle threads. However, having a remove method that could remove the current thread from the list of busy or idle threads would be very convenient. Instead of modifying removeMe() to scan through both lists, I created a new class ThreadArrayList, which extends ArrayList, and a new method, remove():

```
public ThreadElement remove() {
  ThreadElement elem = new
    ThreadElement(Thread.currentThread());
  int i = indexOf(elem);
  if(i == -1)
    return null;
  return (ThreadElement)remove(i);
}
```

I could have created two removeMe() methods (such as removeMeFromBusy() and removeMeFromIdle()), but the subclassing approach is more object-oriented. The busy and idle threads are now instances of the ThreadArrayList class instead of the ArrayList class. To support the call:

```
int i = indexOf(elem);
```

I overrode the equals() method in the ThreadElement class like this:

```
public boolean equals(Object o) {
  if( o instanceof ThreadElement ) {
    return(((ThreadElement)o)._thread ==
      _thread);
  }
  return(false);
}
```

This equals() method will be called by the implementation of the indexOf() method in the ThreadArrayList that it inherits from the ArrayList class, shown here:

```
public int indexOf(Object elem) {
  if (elem == null) {
    for (int i = 0; i < size; i++)
      if (elementData[i]==null)
      return i;
  } else {
    for (int i = 0; i < size; i++)
      if (elem.equals(elementData[i]))
      return i;
  }
  return -1;
}
```

With these changes we do not need the removeMe() function in the thread pool anymore. The removeMe() method in the original thread pool is called only when a thread is in an idle state, yet it searches through all available threads. To remove an idle thread in the original thread pool (either because it has been interrupted or it has timed out), the thread pool calls the removeMe() method. Instead, in the improved thread pool, we can simply call the remove() method on the list of idle threads, like this:

```
_idleThreads.remove();
```

In most cases this is much more efficient than the removeMe() method.

Quantifying Improvements

To quantify results, I have included a program called TestPerformance(), which compares the performance difference between the original pool and the improved pool (available online at www.java-pro.com). This program takes six parameters, although the sixth parameter is optional. The first parameter decides which pool to use. (Specifying "old" uses the original thread pool from the October thread pooling discussion.) The second and third parameters to the program are the maximum and minimum number of allowable threads in the pool. The fourth parameter is the maximum allowable idle time for a thread in the pool before it is destroyed, the fifth parameter is the number of jobs to perform for the test, and the last (optional) parameter is used to turn debugging on (if this parameter is "debug").

On my laptop, which has a PII 400MHz processor and 288M of RAM, I used a thread pool with five threads and a two-second maximum idle time. My averaged results are shown in Table 1. Your numbers will vary depending on your machine configuration, and operating system.

Inefficient coding always degrades performance, but, as these examples show, simple remedies can greatly improve results. Still, you must know when to stop. Too much optimization may lead to code that is difficult to understand and maintain. In this case, the thread pool implementation became more compact as a result of the performance improvements; many times, however, optimized code may be longer than the original version, even though each individual execution path may be shorter and hence perform better. So pick the performance battles to fight—and then look for performance victories. ■

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101TM Download the code for this article.

JP00101TM_T Read this feature online. DevX Premier Club membership is required.

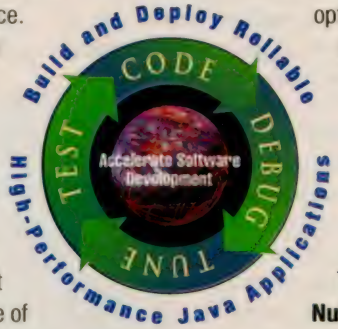
Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Don't Let One Bad Applet Spoil the Whole Application!

TUNE PERFORMANCE ACROSS YOUR ENTIRE DISTRIBUTED APPLICATION WITH NUMEGA® DEVPARTNER® 2.0 JAVA™ EDITION.

Problems with memory use, multi-threading and slow code can "spoil" your application's performance. **NuMega DevPartner 2.0 Java™ Edition** identifies the bad apples in your distributed, web-enabled and e-business applications—quickly and easily.

DevPartner Java Edition is a suite of *performance-tuning* tools that helps you build reliable, high-performance applications and components with Java technology. With support for distributed application development on a range of application servers, servlet engines and web servers,



DevPartner Java Edition allows you to profile, test and optimize components across all tiers of your distributed application so you can measure and improve your application's performance. And, it is the *only* development productivity suite that extends beyond Java code to support web scripting technologies such as JavaScript®, JavaServer Pages® and Active Server Pages.

Get rid of those bad applets and more! Tune your applications for high-performance with **NuMega DevPartner 2.0 Java Edition.**

NEW! Support for Solaris, Linux, Servlets, EJBs and Application Servers.

Accelerate Software Development with NuMega DevPartner 2.0 Java Edition!

Call **1-800-4-NUMEGA** to request a **FREE** copy of our *"Effective Memory Utilization for Reliable, High-Performance Java Applications"* technical paper. Or download the paper today by visiting us at **www.compuware.com/numega**.

1-800-4-NUMEGA
www.compuware.com/numega
30-Day Money Back Guarantee

COMPUWARE®
People and software for business applications™

Copyright © 2000 Compuware Corporation. All rights reserved. Java and all Java-based marks are trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks are the property of their respective owners. ® indicates registration in the United States.

The Payoff of Persistence

Use connection objects to make your socket-connected Web applications more scalable and user friendly

by Cheng-Yaw Chang

The beauty of the Web is that it enables architects to pull together information from many sources to serve its customers. To do this effectively, Web applications must talk to legacy systems on the back end. As long as only one or a few clients engage a single Web server, complications to this process are minimized—references to the connection can be stored in the session object.

However, problems arise when multiple servers are involved, and clients must connect to a legacy system through a socket or TCP/IP channel where the client must be authenticated on the channel. First, the connection is not serializable, so it cannot be passed from one Web server to another, but must be closed and reopened each time. Second, opening a connection to the same server and reauthenticating the client is often very expensive. What we need is a method where the connection can be kept alive and a scheme where the client will always use the same connection. Let's see how two scalable Web architectures use connection objects to maintain persistent connections.

The Problem with Sessions

Most Web servers operate using a "session" architecture. Each client that connects to the server is given a session by either the JavaServer Pages (JSP) or servlets. The session maintains the state of the client connection and client requests. Two prerequisites for information stored in a session are that the data must be serializable so that it can be shared between Web servers, and that the session will time out after the Web client has been inactive for a certain time. This time period is usually configurable, and a notification is sent when the session expires so you can clean it up using HttpSessionBindingListener.

Sharing a session between Web servers results in two major advantages—scalability and load balancing (see Figure 1). First, the client's session state remains the same no matter which Web server routes his or her next request. The system is also load balanced because additional requests can be handled by whichever Web server is available. The client's request thus is fulfilled in a timely man-

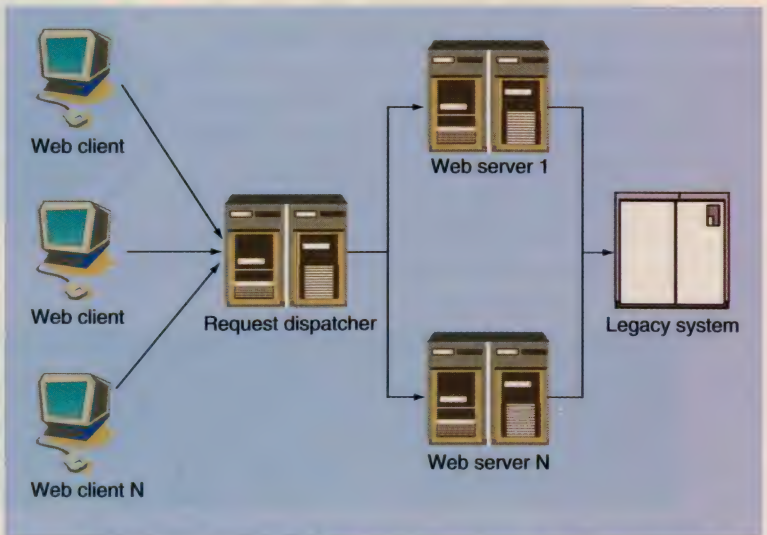


Figure 1 The More the Merrier In this example, multiple Web clients talk to multiple Web servers. A centralized request dispatcher routes client requests to servers, depending on the server's load and capacity. However, the client is not always routed back to the same server that opened the connection. This can lead to problems with client authentication—the client needs to be able to get to the original connection.

Cheng-Yaw Chang is a senior consultant with Technology Enablers Inc. in Dallas TX. He has several years of experience working with C++, Java, Workflow, application servers such as WebLogic and WebSphere, and other technologies such as Enterprise JavaBeans, CORBA, and DCOM. He can be reached at chengyaw@fcmail.com.

ner. But this scalability also presents two problems: the connection isn't serializable, so it cannot be passed from one Web server to another but must be closed and reopened; and constantly opening a connection to the same server and reauthenticating the client is very expensive.

Consider the scenario shown in Figure 1. A centralized request dispatcher routes client requests to Web servers based on the load and capacity of each server. The request dispatcher is load balancing the system, giving the next request to the least busy server. However, the Web client will not always be routed back to the same Web server where it initially opened the connection. Often, the client needs to be able to get to the original connection no matter through which Web server the client is routed.

Could Enterprise JavaBeans (EJB) help with this dilemma? Most enterprisewide Web servers today also have an application server, which allows the user to place business objects in it. Many organizations serve EJB from within the application server, and trying to use EJB technology to open connections to legacy systems is tempting. However, the container maintains the EJB lifecycle, and the container can choose to make the EJB passive at any time. Your connection will then be lost.

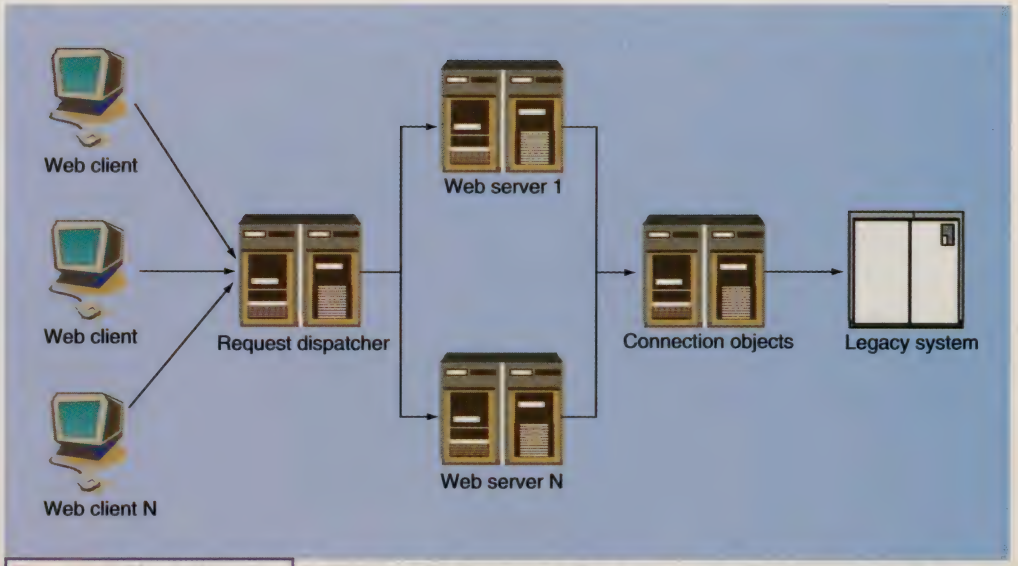


Figure 2 A Simple Location In this example, the request dispatcher routes client requests to the least busy Web server. The Web server then tries to make a remote connection. If it cannot make an initial connection, the Web server places a request with the connection objects server to create the connection. Once the connection is made, communication can begin.

So we need a method where we still have a scalable and load-balanced system, but with nonserializable connections. We need a method that keeps the connection alive—a schema where the system will cache to the open connection so that the client will remain on that connection.

You can persist an open connection in two ways. You can use a location agent that will locate the appropriate open connection such as CORBA, RMI, or JNDI so that client requests can be routed to that connection, or you can use sticky load balancing. In sticky load balancing, user requests are always routed to the same Web server (client to server affinity). First, let's discuss the use of a location agent or service.

Location, Location, Location

Figure 2 shows a simple scenario. The request dispatcher receives a service request and routes it to the least busy Web server using some load-balancing technique. The Web server that receives the request uses a lookup method to connect to a remote connection. Using that connection, the Web server communicates with the back-end system and performs the needed task. If an initial connection is not found, the Web server places a request with the connection objects server to create one, and communication begins on the newly opened connection.

Using this kind of mechanism has several advantages. First, the architecture is tiered. Tiering the system will allow a higher throughput, making the system more scalable. Second, load balancing is easy to do. The request dispatcher will not be a bottleneck when it processes requests. Third, you can use EJB in this architecture (although I wouldn't recommend it). A stateless EJB *can* perform the lookup and fulfill the client request on the found connection. However, the container can make the EJB passive at any time, since the connection is not kept within the EJB. Finally, with this architecture, you can grow the number of Web servers dynamically as needed.

This mechanism also has some disadvantages. The first consideration is cleanup: How will the open connection know when to clean itself up? Second, an additional network lookup could be expensive. Third, in the long run, scaling this system will be harder. The Web servers can scale, but the connection objects server can't scale as easily. Additional coding will have to be done to find objects on the network so that we return to the same user connection every time. Moreover, the connection objects server can be a bottleneck if it services a long request, because all other requests that come in will have to wait—and a slowdown in the connection objects server would mean a systemwide slowdown. Finally, in this system, multiple points of failure are possible.

Sharing a session between Web servers results in two major advantages: scalability and load balancing

If scaling were required on the connection objects repository, then a locator service must be installed to manage the distributed connection objects. Figure 3 shows such a scenario. When a request arrives at a Web server, the server looks up the connection object in the locator service. The locator service returns a reference to the connection object, the Web server connects to it, and the communication starts.

This system has the added costs of the locator-service hardware and the additional programming effort. Additionally, although this system has only a single point of failure (the locator object), its failure could bring the entire system down. (Introducing a hot failover server could solve this problem.)

To make this architecture work, the connection objects have to be wrapped in an interface that can be invoked remotely. This means that the connection objects must have either a CORBA or RMI interface, so that they can be serialized and passed from server to server. (JNDI, in this case, will be the repository that we can store the connection in. In an application server such as WebLogic or WebSphere, we can use JNDI

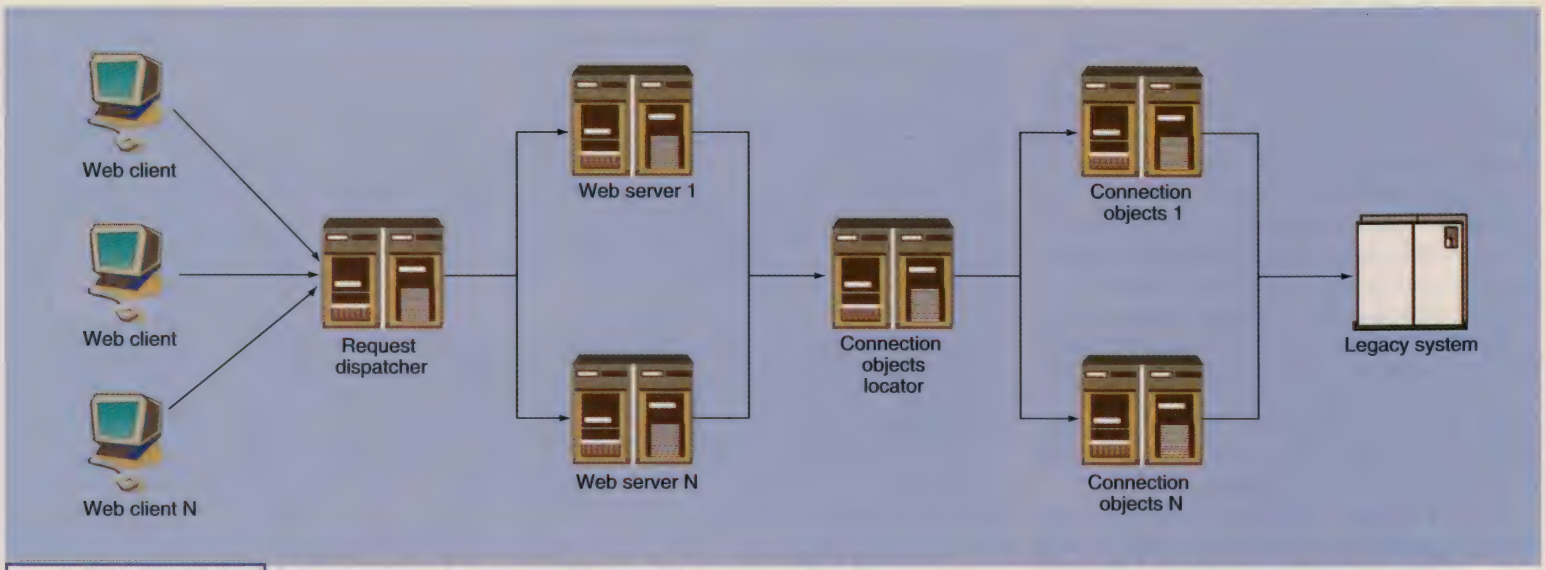


Figure 3 At Your Service In this scenario, when a request arrives at a Web server, the server looks up the connection object in the locator service. The locator service returns a reference to the connection object. The Web server then connects and starts the communication.

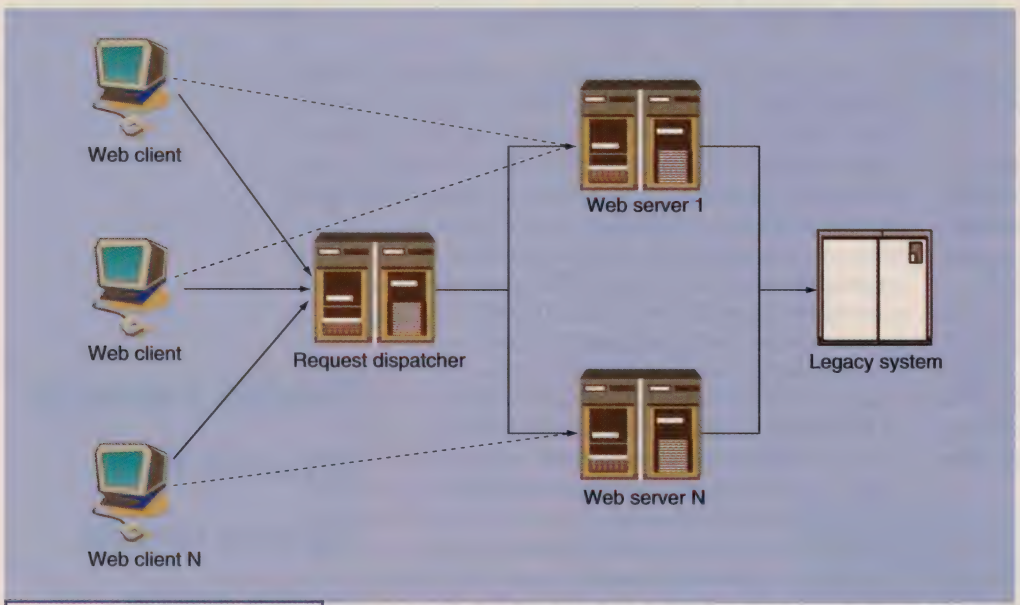


Figure 4 A Sticky Load to Bear In a system that uses sticky load balancing, the request dispatcher routes the user request to the least busy server during the initial connection. The request dispatcher remembers where the user request was routed, and all subsequent user requests are routed to the same server. The dotted lines show the server that each client will always talk to.

to store the connection such that it can be retrieved easily by a stateless session bean.)

You can also use EJB in this architecture, which would give additional scalability to the Web servers. A stateless-session EJB looks up the connection object when a request is received. When the request has been serviced, the EJB releases the reference to the connection object. Then the EJB is released back into the free pool. The connection object can be stored in the JNDI tree. Saving the connection in the JNDI tree will ensure that the connection is not broken.

A Sticky Situation

In a system that employs sticky load balancing, we make full use of the request dispatcher, which directs requests from the same user to the same Web server (see Figure 4). Therefore, if a user logs into server #1, server #1 services all subsequent requests from that user. The connection object in this architecture is kept in the Web session object.

This architecture has several advantages. The system is very scalable—more Web servers can be added as needed. If there is only one Web server, you don't need even the request dispatcher server. Further, the request dispatcher will not be a bottleneck because it merely redirects traffic to another server (thus handling load balancing). This system requires significantly less programming work than the location service method (especially with EJBs)—and consequently less overhead—and it has fewer points of failure. Finally, there is no session sharing, which cuts down the amount of network traffic.

This method has only one real disadvantage: the request dispatcher will have to be configured, and the smart routing causes a slight delay in the servicing of the requests. (Configuring the request dispatcher only needs to be done once and it should not really be listed as a disadvantage. The delay in the request is the only real disadvantage.)

The decision to use either of these techniques is based on the Web server and request dispatcher that your company decides to purchase. Many Web servers and request dispatchers are available on the market today, and not all of them have the same capabilities. Most Web servers now have an application server that supports EJB technology, but do not be blinded by this functionality. There is a time and place for it, and the caching of nonserializable connections is not necessarily the time or place. The requirement to remember is that the connection should be kept alive and the client should always use the same connection. Either of these two scalable Web architectures can do this. It will be up to your circumstances to make the choice that's right for you. ■

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to this related resource.

JP0101CC_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Borland

Make the jump to JBuilder 4

The sky's the limit with the #1 rated Java development tool.

JBuilder 4's cross-platform environment brings you rapid, enterprise-strength Java application development and the most comprehensive set of Java tools — including new team collaboration, and highly productive Web, EJB and J2EE development.

Borland's JBuilder 4 will take you to the extreme...

where almost anything goes.

Order at shop.borland.com
or call 1-800-252-5547

www.borland.com

Send Users a Browser Message

Got user error problems? Use the servlet session API and JavaScript to display a useful message during a long-running servlet

by Timothy Eden

One of the reasons I enjoy creating Java-based Web applications is because I enjoy solving puzzles. Recently I ran into one of these puzzles while working on a Java servlet application—how to display a message in the browser while a long-running servlet executes. How could I solve this?

My project required an HTML form to pass information to a servlet that executed a SQL statement using JDBC. The servlet gathered the information, processed it, and then returned the output to the browser. This no doubt sounds familiar to anyone who works with servlets. However, one big problem arose with my servlet—processing the SQL statement took 45 seconds, leaving my business users to believe that the application had stopped working. The user then pressed the browser's refresh button, thus increasing the processing time of the servlet.

I know what you're thinking. Doesn't the browser's icon in the upper right-hand corner continue turning while the servlet is executing? Shouldn't this icon indicate to the user that the application is not finished executing? Yes, it does, and yes, it should. However, many individuals in the business community still are not browser educated. That's why I needed a message to be displayed to inform my users that the application was still processing.

So how do you put a message on the browser while the servlet executes? After all, when you send the information from an HTML form to the servlet, there isn't an immedi-

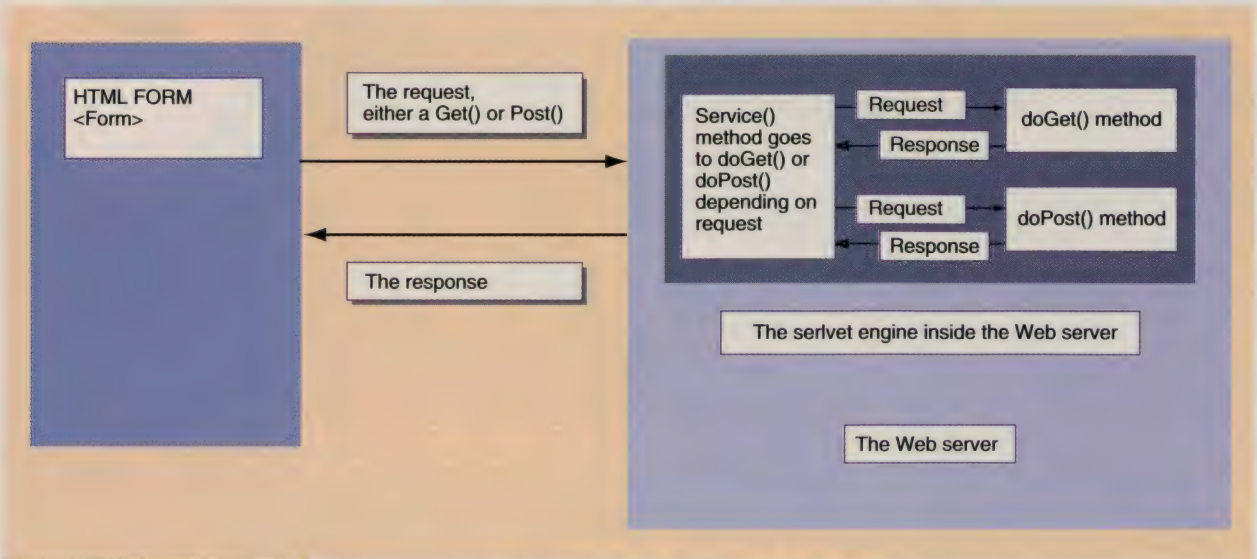


Figure 1 Positive Interaction This figure shows the basic interaction between an HTML form and servlets.

Timothy Eden is a lead developer for ExtremeStyle Inc in Jacksonville, Florida, and a graduate of Auburn University. He can be reached at tim@extremestyle.com or timothyeden@mediaone.net.

ate Web page to display a message while the servlet executes. Or so I thought before I discovered a technique to solve my problem. The answer lies in the use of the servlet session API and JavaScript.

If You've Never Used Servlets

In a typical HTML form-to-servlet interaction, the user fills out the HTML form and then, using a Get() or Post() method, submits the information to the Web server (see Figure 1). The Web server, running a servlet engine, processes the information in the servlet's doGet() or doPost() method. (This is a very simple explanation of servlets; for a complete explanation, go to www.apl.jhu.edu/~hall/java/Servlet-Tutorial/ or http://archive.coreservlets.com.)

One of the problems in Web development is maintaining state. Whenever you go from one Web page to another, any information that you entered on the previous page is forgotten. The previous page cannot maintain the value, or "state," of any information you entered.

This problem has several solutions. One is placing a cookie on the user's machine; another is passing the information with the URL (looks something like www.anything.com?LastName=Eden&FirstName=Amy). Both of these processes work, but each has a drawback. A user can set his or her browser not to accept cookies, and information passed in a URL can go only up to 256 characters. But if you're using servlets, you have another option—a session object.

A session is the interaction between a Web server and an individual user for a specified amount of time. The servlet session object allows you to store and retrieve information on the server that is related to that individual user, such as their userid and password. Then, that information can be retrieved anywhere in your application for the duration of the session.

For a servlet, the javax.servlet.http.HttpSession class allows you to create a session for each user and store and retrieve Java objects such as strings or vectors (see Listing 1). Or you can create a session in a servlet, then store and retrieve a vector using two servlets (see listing online at www.java-pro.com). The session object doesn't reside on the server forever. If a session object is not accessed within a certain amount of time, the session expires—in other words, the session object is removed from the server. Depending on your servlet engine provider, the default

Because my users couldn't tell that the servlet was still processing, they kept hitting the Refresh button, which prolonged the event

Listing 1 The Object of This The javax.servlet.http.HttpSession class lets you create a session for each user and store and retrieve Java objects.

```
import javax.servlet.*;
import javax.servlet.http.*;
public class YourServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req,
    HttpServletResponse resp)
    throws ServletException, IOException
  {
    //create the session for the user.
    HttpSession session1 = req.getSession(true);
    //store your name in a session variable-MyHighSchool
    session1.putValue("MyHighSchool","WarnerRobinsHigh");
    //to retrieve the session variable "MyHighSchool"
    String s = session1.getValue("MyHighSchool");

  }  //end of doGet
}  //end of YourServlet class
```

Listing 2 Show Me the Message This is the servlet that uses JavaScript and session variables to display a message while ProcessingServlet is executing.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Vector;


public class ShowMessage extends HttpServlet
{

  public void doGet(HttpServletRequest req,
    HttpServletResponse resp)
    throws ServletException, IOException
  {
    //get the first and lastname passed in from Begin.html
    firstname = req.getParameter("FirstName");
    lastname = req.getParameter("LastName");

    HttpSession session = req.getSession(true);
    //put the first and lastname into session variable
    session.putValue("firstname", firstname);
    session.putValue("lastname", lastname);

    PrintWriter out = resp.getWriter();
    //set content type and other response header
    //fields first
    resp.setContentType("text/html");

    out.println("<HTML>");
    out.println("<SCRIPT LANGUAGE='JavaScript'>");

    out.println("function GoToDifferentServlet ()
      {this.location=\"/servlet/ProcessingServlet\"};");
    out.println("timeout=window.setTimeout(
      \"GoToDifferentServlet ();\", 3000);");

    out.println("</Script>");
    out.println("<Body bgcolor=\"white\" >");
    out.println("<Center>");
    out.println("<BR>");
    out.println("<font Size=+2 color=\"blue\">");
    out.println("<br>The Processing Servlet is executing -
      War Eagle Go Auburn!");
    out.println("<br>Please wait until a confirmation
      message appears.");
    out.println("</fONT></BODY></html>");

  }   //end of doGet
}   //end of ShowMessage Servlet
```

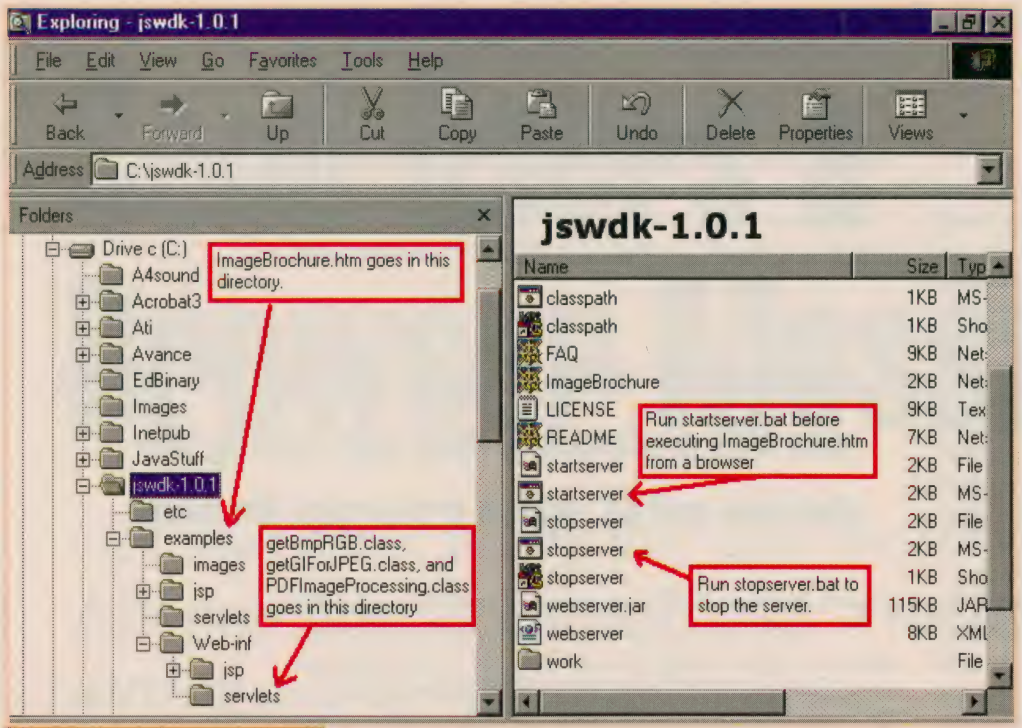


Figure 2 The JSWDK Structure This is what the JSWDK directory structure will look like after you extract the files from the JSWDK1_0_1-WIN.ZIP.

timeout is normally 30 minutes. You don't have to wait for the session to expire if you want to eliminate a session object. The Session API gives you two options to destroy a session object—the setMaxInactiveInterval() and Session.invalidate() methods. The setMaxInactiveInterval() method is available only in the Servlet 2.2 API. And you can destroy a user's session if it has not been accessed for at least 10 minutes (see listing online).

Those are the basics for the first part of the solution—the session object. Now let's talk about the second part of the solution—JavaScript.

JavaScript is the Key

JavaScript is the key to displaying a message Web page while your servlet is processing. You'll use JavaScript's OnLoad event and setTimer() method to accomplish this task. The Onload event executes the setTimer() method once the HTML page is loaded into the browser. The setTimer() method then fires off another JavaScript method that redirects the browser to start the processing servlet. The important point to remember is that the informational page will continue to be displayed

JAVAPro Product Showcase

Bar Code JavaBeans & Fonts

JavaBeans, Applets, Barcode Fonts and HTML embedded fonts for Code 39, POSTNET, Code 128, UPC, PDF417, OCR, MICR and many others for integration into your application. We also offer barcode UFLs for Crystal Reports, MS Office Macros and Java Source Code.

IDautomation.com Inc.
http://www.IDAutomation.com
admin@idautomation.com

This is your Showcase Section for java tools and applications. If you are interested in promoting your tool or product to Java developers, please call 650.833.7100.

FTP FAWCETTE TECHNICAL PUBLICATIONS

Java is a registered trademark of Sun Microsystems, Inc.

JAVAPro

Subscribe Now!

Java Pro is the progessional developer's independent source for information about building serious Java applications.

Java Pro takes you beyond the hype by delivering hands-on, code-intensive, product-oriented information that helps you harness Java's unique capabilities in your enterprise-wide development efforts. Each month, Java Pro tackles tough development issues and provides unbiased, no-nonsense coverage of the latest tools on the market.

Call today to subscribe to Java Pro. You'll pay the low rate of $29.95 for a full year-thirteen issues in all. That's a savings of more than 61% off the newsstand rate. To subscribe now call 800-848-5523. Outside the U.S. call 650-833-7100. Or, order online at www.java-pro.com.

FTP FAWCETTE TECHNICAL PUBLICATIONS

Fawcette Technical Publications
209 Hamilton Ave.
Palo Alto, CA 94301

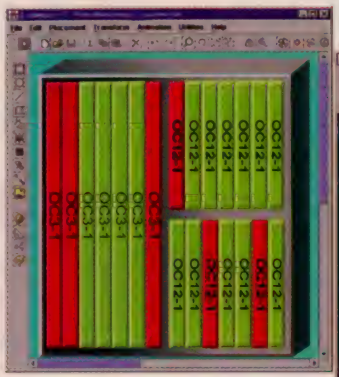
Java is a registered trademark of Sun Microsystems, Inc.

Accelerate your Java™ Graphics Development Today

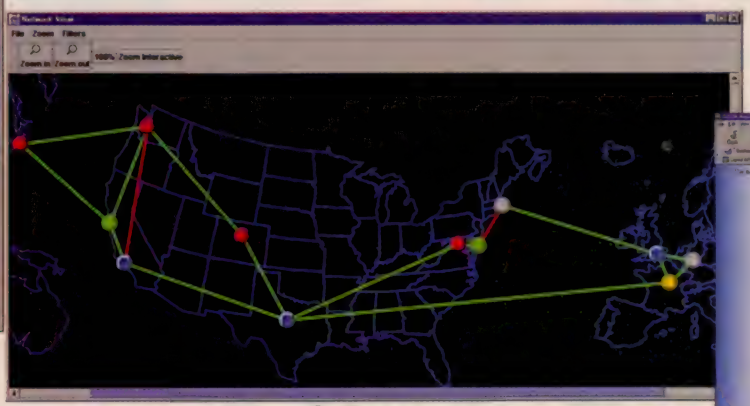
Ask about our free 30-day evaluation program

JLOOX

JLOOX is the best-in-class graphics toolkit for Java 2 developers. Based on Java2D, JLOOX provides a high-level 100% Java API that offers breakthrough performance and productivity gains for developers of graphics intensive Swing applications. It is ideal for applications requiring highly graphical displays such as networking and telecommunications,intelligent transportation systems, process control and simulation.



Element Management



Network Management

Instrumentation

Unique advantages:

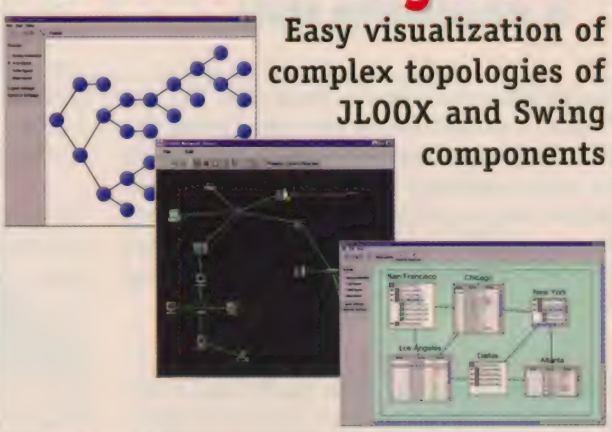
- Includes an interactive editor for rapid creation and animation of graphic screens
- High-level intuitive API decreases development time
- Pre-built collections of objects provide powerful data-visualization primitives
- Offers the rendering power of Java2D
- Includes the award winning JClass Chart from KL Group
- Can be used with any IDE supporting Java 2
- Applications created with JLOOX have no runtime, royalty or deployment fees

JLOOX GIS

Ultra-fast Java mapping components

JLOOX Layout

Easy visualization of complex topologies of JLOOX and Swing components



Accelerating Graphics Development™

LOOX Software Inc.

JAVA Pro 2000 Readers' Choice Winner

1860 El Camino Real, Ste. 306, Burlingame, CA 94010 • T:650-552-9310 F:650-552-9307 • www.loox.com

Copyright (c) LOOX Software Inc. 1999. LOOX is a registered trademark and JLOOX is a trademark of LOOX Software Inc. All other trademarks are the property of their respective owners.

Listing 3 The Process Takes Time This servlet simulates a 25-second processing time. It might take this long for the servlet to execute a SQL statement.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpUtils;
import java.util.Hashtable;

public class ProcessingServlet extends HttpServlet
{
  String firstname,lastname;

  public void doGet(HttpServletRequest req,
    HttpServletResponse resp)
    throws ServletException, IOException
  {

    //open the session object--give the session object
    //any name you want. I choose session10 to show that
    //the object doesn't have to be called session.
    HttpSession session10 = req.getSession(true);

    //get the firsname and lastname stored in this
    //session object
    firstname = (String)session10.getValue("firstname");
    lastname = (String)session10.getValue("lastname");

    resp.setContentType("text/html");

    PrintWriter out = resp.getWriter();

    out.println("<HTML>");
    out.println(
    "<HEAD><TITLE>ExampleServlet Output</TITLE></HEAD>");
    out.println("<BODY>");

    //to do: your HTML goes here.

    long ntime = System.currentTimeMillis();
    //continue to loop for 25 seconds to simulate a
    //long processing servlet
    while( ntime + 25000 > System.currentTimeMillis())
    {
      //the System.out.println was put in here so you could
      //see some processing going on.
      System.out.println(System.currentTimeMillis());
    }
    out.println("<BR><Center>servlet is finished");
    out.println("<BR>First Name is "+firstname);
    out.println("<BR>Last Name is "+lastname);

    out.println("</BODY>");
    out.println("</HTML>");

  }   //end of doGet method
}   //end of class
```

in the browser until the processing servlet finishes executing. In my case, the processing servlet took 45 seconds to complete.

When the processing servlet finishes, it sends HTML output to the browser, thus replacing the informational page with a new Web page. One thing needs to be mentioned here about the setTimer() method in JavaScript. The setTimer() method makes any other JavaScript function execute in a loop based on the amount of milliseconds you supply to setTimer() arguments. However, since we are using setTimer() to excute a function that redirects the browser to another page, the setTimer() method executes only once (see Figure 1).

One of the problems in Web development is maintaining state

Solving the Problem

Before we look at the code, you'll need a Web server that contains a servlet engine to run any of the servlets. If you don't have access to one, I recommend using the JavaServer Web Development Kit (JSWDK) from Sun. You can download this software from www.javasoft.com/products/jsp/download.html. Once you install the JSWDK, place your servlets in the examples/Web-inf/Servlets directory and your html files in the wwwww directory. Figure 2 shows what the directory structure looks like after installing the JSWDK. Now let's look at the code for Begin.html, which sends information to the intermediate message servlet ShowMessage:

```
<HTML>
<BODY>
<FORM ACTION=
  "/servlet/ShowMessage" METHOD="POST" >
<P>FirstName<INPUT TYPE="TEXT" NAME=
  "FirstName" SIZE="25"></P>
<P>LastName<INPUT TYPE="TEXT" NAME=
  "LastName" SIZE="25"></P>
<P><INPUT TYPE="SUBMIT" NAME=
  "Submit" VALUE="Submit">
</FORM>
</BODY>
</HTML>
```

This page has a form with just three elements—a textbox for a first name, a textbox for a last name, and a submit button. When you press the submit button, the HTML form submits the values for the first and last name to the display message servlet ShowMessage (see Listing 2). This servlet accomplishes three tasks. First, the values for the first and last name are stored in servlet session variables. Second, the message "The servlet is executing" is output to the browser. Last, this JavaScript code executes.

```
function GoToDifferentServlet()
  {this.location="http://localhost:8080/
  examples/servlet/ProcessingSerlvet"};
timeout=window.setTimeout(
  "GoToDifferentServlet();", 3000);
```

This Javascript code redirects the browser to the servlet ProcessingServlet 3,000 milliseconds or three seconds after ShowMessage loads into the browser. The message "The Processing Servlet is executing—War Eagle Go Auburn!" displays in the browser until the servlet ProcessingServlet finishes executing.

The last piece to the puzzle is the processing servlet (see Listing 3). I've put a loop inside the code to simulate a servlet executing a SQL statement that takes a while to finish. Once the loop finishes, the information supplied to the first and last name textboxes in Begin.HTML is output.

That's all there is to it. So the next time you have a set of impatient users, you might want to send them a message. It's easier than it looks, and it can save a lot of time and aggravation. ■

Go Online

Use these DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101TE Download the code for this article.

JP0101TE_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Become a charter subscriber NOW!

PROGRAM BETTER AND FASTER. Subscribe Now.

$19.95 12 issues annually

Avoid 5 Common Release Bugs
Visual C++
Developers Journal
8 Design Tips for Building COM Interfaces
Delve Into COM+ Events
Throw and Catch COM Exceptions
Manipulate 3-D Objects

Name

Address

City State Zip

E-mail

O Payment Enclosed O Bill Me O Charge My: O Mastercard O VISA O AMEX

Card No. Exp. Date

Signature

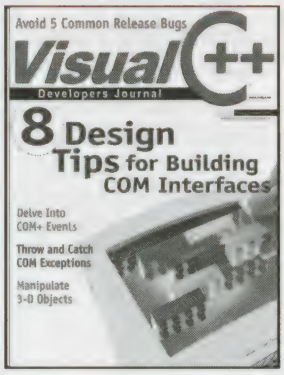
One year is 12 monthly issues. Please allow 4–6 weeks for delivery of first issue. International subscriptions must be payable in U.S. dollars plus postage. Canada and Mexico add $18/year, all other countries add $44/year for airmail delivery. Canadian GST included.

4929

Become a charter subscriber NOW!

PROGRAM BETTER AND FASTER. Subscribe Now.

$19.95 12 issues annually


Avoid 5 Common Release Bugs
Visual C++
Developers Journal
8 Design Tips for Building COM Interfaces
Delve Into COM+ Events
Throw and Catch COM Exceptions
Manipulate 3-D Objects

Name

Address

City State Zip

E-mail

O Payment Enclosed O Bill Me O Charge My: O Mastercard O VISA O AMEX

Card No. Exp. Date

Signature

One year is 12 monthly issues. Please allow 4–6 weeks for delivery of first issue. International subscriptions must be payable in U.S. dollars plus postage. Canada and Mexico add $18/year, all other countries add $44/year for airmail delivery. Canadian GST included.

4929

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

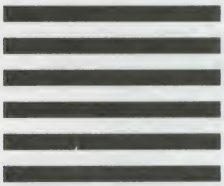
POSTAGE WILL BE PAID BY ADDRESSEE

Visual C++
Developers Journal
P.O. Box 59289
Boulder, CO 80323-9289

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 317 PALO ALTO, CA

POSTAGE WILL BE PAID BY ADDRESSEE



Visual C++
Developers Journal
P.O. Box 59289
Boulder, CO 80323-9289

Code Faster, CODE Smarter

Your Technical Guide to Faster, Better VC++ Code

You're an experienced, savvy Visual C++ professional, and *VCDJ* is your indispensable tool for faster, smarter code!

There's no better independent guide to hardcore, professional Visual C++ programming. You can't afford to miss one issue, or even one article, of *VCDJ's* practical advice for corporate development. You'll find timesaving tips and proven techniques for building server-side components, client executables, Web apps, or anything Visual C++ can create. Each issue will bring you hands-on, code-intensive articles that help you:

- Turbocharge your Visual C++ apps' performance
- Choose the right features and functions to balance speed and functionality
- Expose the right COM interfaces in the right way
- Master the latest Visual C++ techniques and tools
- Control memory allocation like a pro
- Take advantage of new programming paradigms
- Access data fast
- Harness the power of XML

Don't miss a single issue.

Visual C++ Developers Journal

Order online

www.vcdj.com

Order by phone: 1-800-848-5523

Outside the U.S. : 1-650-833-7100

VCDJ is another high-quality, how-to publication from Fawcette Technical Publications

VISUAL C++ IS A REGISTERED TRADEMARK OF MICROSOFT CORP. ALL OTHER PRODUCT NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.

GO FROM E-CONCEPT TO E-BUSINESS IN NO TIME FLAT.

Featuring

VisualCafé | StructureBuilder | TopLink | Macromedia Dreamweaver

We've acquired and licensed some of the best technologies on the planet
and created an integrated suite that can cut weeks,
maybe even months, off your application development schedule.

Introducing WebGain Studio.™

It's a complete, end-to-end, browser-to-database Java development solution.
And from design to code to deployment, there's nothing more powerful or productive.

Check it out today at **www.webgain.com**.

webGAIN™

©2000 WebGain, Inc. WebGain, WebGain Studio, VisualCafé, StructureBuilder and TopLink are registered trademarks of WebGain, Inc.
Dreamweaver is a registered trademark of Macromedia, Inc.

Javatecture

Stupid Browser Tricks with JavaServer Pages

Add flexibility and control to your Web pages with JSP—they generate just the JavaScript you need to support different browsers

BY JAMES W. COOPER

By now you've read enough articles about JavaServer Pages (JSP) to understand fairly well what you might use them for. They are more powerful than CGI, more portable than ASP, and a lot easier to use than either of these other client/server methods. They can probably even leap tall buildings in a single bound! Remember that a JavaServer Page usually consists of some page script that runs on the server and performs some computations or makes some decisions and generates pure HTML output. The script usually references some Java classes in a bean on the server and uses them to make the computation.

Usually, you think about using JSP for looking up data based on some input parameters and displaying that data in the generated browser page. For example, you might display the results of a search engine's search, or a list of products and prices, or the names and addresses of people in a certain category you specify.

However, you can use JSP to control more than the data you display. You can also control how the browser displays it. To see how this can be done, let's review some of the characteristics of JavaScript. Yes, I know that JavaScript isn't Java, but we will end up using Java, too, so trust me on this.

A Bit about JavaScript

As you probably know, JavaScript is an interpreted language that is executed as part of a Web page when it is loaded and when you perform certain actions on the form's controls. The most common use of JavaScript is in interacting with HTML form fields. When your page's user fills in some fields and clicks on Submit, you can check the page for errors before you submit it to your server.

When you submit an HTML form, you typically do it by specifying the next form to load in the action field of the form.

```
<form action= "search.jsp">
<input type=submit name="search" value="Search">
```

If you do not specify a new JSP page, the same one gets reloaded with whatever changes the script specifies after the data in the form is processed. A more powerful way to submit pages is to have the

Listing 1 Splash Your Browser Rather than using JavaScript to encapsulate differences in browser type, you can use JSP to generate the page code you need for a particular browser. This splash screen page figures out what kind of browser you have.

```
<%@ page language="java"
  import="SMenu.MenuBean" %>
<jsp:useBean id="smenu" scope = "session"
  class="SMenu.MenuBean"/>
<jsp:setProperty name="smenu" property="*"/>

<html><head>
<title>Opener Page</title>
<!------------------------>
<script language ="JAVASCRIPT">
//detects which browser we are using
function showData(form) {
  form.action= "ShowMenu.jsp";
  var browser = navigator.appName;
//sets string into hidden input variables
  form.browserType.value = browser;
  form.browserVersion.value = navigator.appVersion;
  form.submit();
  }
</script>
</head>
<!------------------------>
<body>
<center>
<h2>Welcome to the Opener</h2>

<form>
<input type=button name="enter"
  value="Enter site"
  onClick="showData(this.form)">
<INPUT TYPE="HIDDEN" NAME="browserType" value="foo">
<INPUT TYPE="HIDDEN" NAME="browserVersion" value="foo2">
</form>

</center>
</body>
</html>
```

Listing 2 Beans on the Menu The browser name and version number values are sent to the JSP bean named MenuBean, where the two methods in this listing are called automatically.

```
package SMenu;
public class MenuBean {
  private String browserVersion;
  private String browserType;
  //saves the browser name
  //saves the browser name
  public void setBrowserType(
    String btype) {
    browserType = btype;
  }
  public String getBrowserType() {
    return browserType;
  }
  //saves the browser version
  public void setBrowserVersion(
    String bversion) {
    browserVersion = bversion;
  }
  public String getBrowsVersion() {
    return browserVersion;
  }
```

button call a JavaScript function, which then itself executes the submit, usually after some input validation.

```
<input type=button name="search"
 value="Search"
 onClick="searchData(this.form)">
```

The corresponding JavaScript function could be as simple as:

```
function showData(form) {
 form.action= "search.jsp";
 form.submit();
}
```

If you want to make an interesting client page without using Java, you can frequently do it using just JavaScript. For example, you might have a set of floating menu items along the side of your page that you want to highlight when the mouse moves over them. Floating items are placed in a special division or layer that you can position by pixel and even move while the page is displaying. In Netscape browsers, you position this floating object inside <layer> tags.

```
<layer name="ourmenu" left=50 top=100>
Hi there
</layer>
```

Layers are accessible through a layers[] array of the Document object, and the elements of the array are in the same order as they appear in the document.

```
Document.layers[0].top = 75;
```

You can also make layers highlight and unhighlight when the mouse moves over them by setting the onmouseover property to a function to be called.

```
var mLayer = new Layer(menuWidth);
mLayer.onmouseover = hiLight;
mLayer.onmouseout = unLight;
```

Resources

JavaScript—The Definitive Guide, David Flanagan (O'Reilly, 1998)

Dynamic HTML—The Definitive Reference, Danny Goodman (O'Reilly, 1998)

Java Design Patterns: A Tutorial, James Cooper (Addison Wesley Longman, 2000)

```
//————————————
function hiLight() {
 this.bgColor = menuForeColor;
}
//————————————
function unLight() {
 this.bgColor = menuBackColor;
}
```

While layers are not part of any W3C standard, as far as I am aware, they are definitively a part of Netscape browsers. (This changes somewhat in Navigator 6, but at this writing the final version has not been released.)

If we want to draw menus for any browser, we need to encapsulate their differences—a good way is to use JSP to generate just the page code we need for a particular browser

In Explorer, you can assign dynamic positions to any elements, and you typically use a tag to create a block to be positioned.

```
<div id="ourmenu"
 style="position:absolute">
Hi there
</div>
document.all.ourmenu.style.pixeltop = 75 ;
```

If you want to have an anchor tag that is not underlined and changes color with a mouseover, you set it using style sheet tags:

```
<style>
a:hover{
 color:#ff00ff;
 font-weight:bold;
 background-color:#FFFF00;
}
```

"Best cross-platform development tool we saw bar none."

–David Aubrey and Oliver Rist
Internet Week, October 2, 2000

"CodeWarrior will be a powerful asset to those who are breaking into the Internet device space."

–Tim Fielden
InfoWorld, May 8, 2000

"I immediately found the environment intuitive and easy to use."

–Tim Fielden
InfoWorld, May 8, 2000

"With CodeWarrior, you can really create your own (custom) environment."

–David Aubrey and Oliver Rist
Internet Week, October 2, 2000

"All of these features combined means more time providing functionality and faster time to market for deploying PersonalJava applets and applications."

–Sue Spielman
JavaPro, July 24, 2000

"Metrowerks CodeWarrior for Java warrants a serious look from device developers."

–Sue Spielman
JavaPro, October 2000

"Entirely open IDE in order to develop rock-solid, wizard-less and cross-platform code."

–David Aubrey and Oliver Rist
Internet Week, October 2, 2000

"We're awarding this Best of Breed to the IDE that best backs up Java programmers who know their code – that tool is definitely Metrowerks CodeWarrior."

–David Aubrey and Oliver Rist
Internet Week, October 2, 2000

"CodeWarrior IDE is a solid environment with a great debugger and other integrated IDE tools."

–David Aubrey and Oliver Rist
Internet Week, October 2, 2000

"The debugging capabilities really made the tool shine."

–Tim Fielden
InfoWorld, May 8, 2000

It seems like everyone but your Mother is raving about CodeWarrior for Java™ 6.0.

(But she always liked you better anyhow.)

CodeWarrior for Java Version 6.0 supports J2SE, PersonalJava and J2ME open standards for programming Java applications for desktop and wireless devices.

Order your copy today at:
www.metrowerks.com

metrowerks
Software Starts Here

Listing 3 Different but Same The two files, NSMenus.js and IEMenus.js, are an encapsulation of the menu JavaScript code. They have identical methods, but different implementations.

```
<script language="JavaScript">
function setMenus() {
  menuBackColor = "#ffffff";
  menuForeColor = "#FFFF00";
  menuHighColor = "#ffc000";
  menuX = 50;
  menuY = 100;
  menuWidth = 200;
  menuHeight = 20;
  menuBottom = 300;

  var index = 0;
  addMenu("Lab Software",
    index++, "http://labsoftware.com");
  addMenu("Connecticut Swimming",
    index++,"http://ctswim.org");
  addMenu("Amazon",
    index++, "http://amazon.com");
  addMenu("US Postal Service",
    index++, "http://usps.gov");
  drawMenus();
}
</script>
<style>
a:hover{color:#ff00ff;
  font-weight:bold;
  background-color:#FFFF00; }
</style>

<style>
A {text-decoration:none}
</style>
<body onLoad="setMenus()">

<h1> Favorite Sites</h1>
</body></html>
```

```
</style>
<style>
A {
 text-decoration:none
}
</style>
```

This latter text-decoration style declaration works for both Netscape and IE.

Supporting Multiple Browsers

If we want to be able to draw our menus irrespective of which browser we use, we need to encapsulate these differences. Typically, you write a lot of JavaScript with if blocks in it.

```
if (NetscapeBrowser)
 do this;
else
 do that;
```

A better way is to use JSP to generate just the page code we need for a particular browser. We start with a splash screen page that figures out what kind of browser we have (see Listing 1).

The two hidden input variable fields are set to the browser name and version number when you click on the button labeled Enter Site. Then these two values are sent to the JSP bean, named MenuBean, where the setBrowserType() and setBrowserVersion() methods in Listing 2 are called automatically.

The other thing we do is encapsulate the generation of the menu JavaScript code into two files with identical methods but different implementations, called NSMenus.js and IEMenus.js. They each have addMenu() and drawMenu() functions and highlight and unhighlight functions. Our main program is shown in Listing 3.

The critical part, however, is the JSP part that decides which menu code to include. We need to have a statement that gets the correct menu from the JSP bean:

```
<script src=
 <% out.println("\"" +
 smenu.getBrowserMenu() +
 "\""); %>
 language="JavaScript">
</script>
```

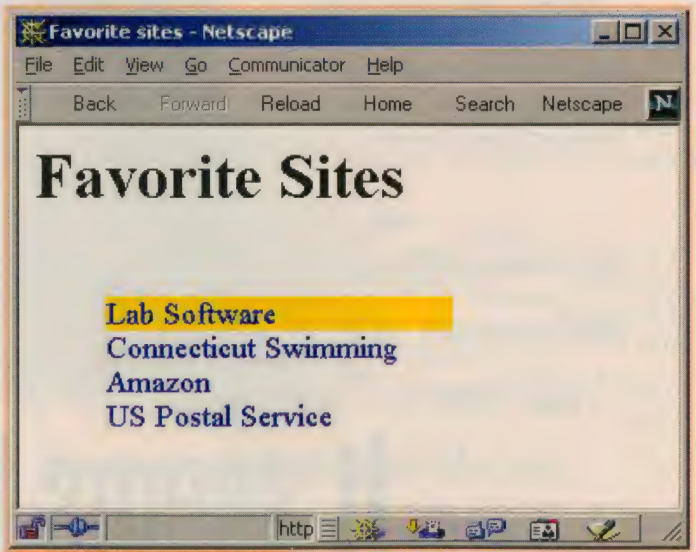


Figure 1 Here's Your Menu The IE version has only minor differences in font size from this Netscape version. You can extend this dynamic JavaScript generation to any level of detail, but with the advantage that no page contains JavaScript code for more than one browser, which makes the code easier to write and maintain. If you need to specify different behavior for other browser versions, you can add as many new versions as you want without tangling up your JavaScript code.

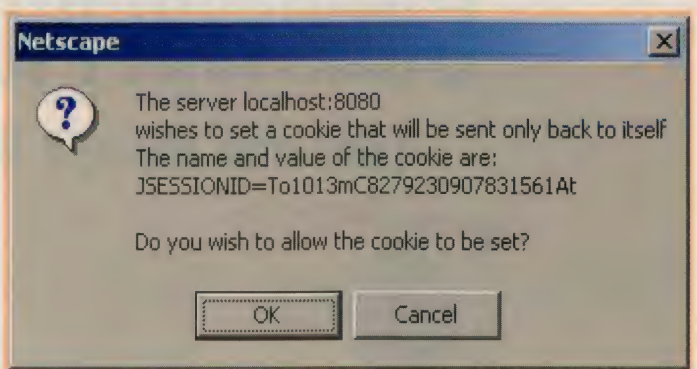

Netscape
The server localhost:8080
wishes to set a cookie that will be sent only back to itself
The name and value of the cookie are:
JSESSIONID=To1013mC8279230907831561At
Do you wish to allow the cookie to be set?
OK
Cancel

Figure 2 Want a Cookie? The IE message is similar to this Netscape version, but you do not have to use cookies for this. You can include these same two hidden fields on each client page and query the bean for their values (which are set as each page is submitted). Simply include the JSP code to generate these two hidden fields with the values filled in.

XML Solutions and Strategies:

Get them here first!

How to Choose an XML Ser
XML
MAGAZINE
.NET=XML
How XML Will Move Microsoft Beyond Windows
Apps to Web Services with SOAP
Serving XML with JavaServer Pages
Expert Voices: .NET's Wascha on XLANG, Hejlsberg on C#
Cocoon Separates Logic from Layout
Microsoft .net

FREE Subscription!

XML Magazine is the only independent publication devoted 100-percent to the latest information about using XML for enterprise-level challenges. Created by the same experts who gave you *VBPJ, Exchange & Outlook* and *Java Pro,* this is the one indespensible XML resource that you can't afford to be without. Published every other month, you'll get practical, hands-on articles, tips and enterprise-level solutions, including:

- Pragmatic techniques for developing with XML
- Cutting-edge analysis of industry trends
- Independent product reviews and new product announcements
- Columns by the industry experts who design the technology
- Complete source code examples

Sign up today for your FREE subscription. Complete the postage-paid card in this issue.

FTP FAWCETTE TECHNICAL PUBLICATIONS

209 Hamilton Avenue, Palo Alto, CA 94301-2500 • Phone: 650.833.7100 • Fax: 650.853.0230

©2000, FTP, Inc. XML is a trademark of MIT. Publisher reserves the right to determine qualifications for free subscriptions.

The corresponding Java in the JSP bean implements that getBrowserMenu() function:

```
public String getBrowserMenu() {
 int i =
  browserType.indexOf ("Netscape");

 if( i >=0 )
  return "NSMenus.js";
 else
  return "IEMenus.js";
}
```

So the generated page code will then contain either:

```
<html>
<head>
 <TITLE>Favorite sites</TITLE>
 <script src="NSMenus.js"
  language="JavaScript">
```

You can use JSP to control more than the data you display. You can also control how the browser displays it

or the corresponding statement for IEMenus.js. The menu page is shown in Figure 1.

Persistence of Values

In our example, we declared that our JSP connection had session score:

```
<jsp:useBean id="smenu"
 scope = "session"
 class="SMenu.MenuBean"/>
```

This means that the values for the browserType and browserVersion variables persist throughout the session. Under the covers, the JSP engine manages this persistence using cookies. To prove this, set your browser to warn you about cookies, and you will get the warning shown in Figure 2.

```
<form>
<INPUT TYPE="HIDDEN"
 NAME="browserType" value=
 <% out.println("\"" +
 smenu.getBrowserType() +"\"");
%>
>
<INPUT TYPE="HIDDEN"
 NAME="browserVersion" value=
 <% out.println("\"" +
 smenu.getBrowserVersion() +"\"");
%>
>
</form>
```

For Netscape, this gets filled in as:

```
<form>
<INPUT TYPE="HIDDEN"
 NAME="browserType" value="Netscape"
>
<INPUT TYPE="HIDDEN"
 NAME="browserVersion" value=
 "4.74 [en] (Windows NT 5.0; U)"
>
</form>
```

and for Internet Explore the page contains:

```
<form>
<INPUT TYPE="HIDDEN"
 NAME="browserType" value=
 "Microsoft Internet Explorer"
>
<INPUT TYPE="HIDDEN"
 NAME="browserVersion" value=
 "4.0 (compatible; MSIE 5.01; Windows
 NT 5.0)"
>
</form>
```

In essence, we have used JSP to create a JavaScript factory. The JSP bean generates just the JavaScript we need for each browser we want to support. In cases where Java is not appropriate for the client page, this gives you a great deal of flexibility in providing JavaScript implementations of your Web pages. ■

James W. Cooper is a computer science researcher and the author of 13 books. His latest, *Java Design Patterns: A Tutorial*, was published by Addison Wesley Longman in January 2000.

Go Online

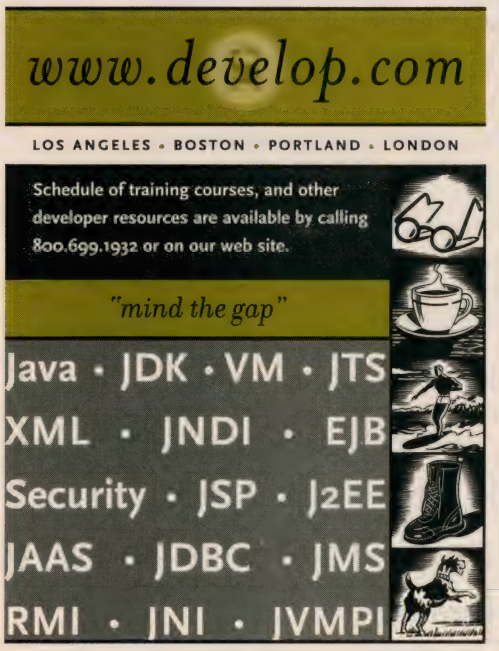
Use the following DevX Locator+ codes at www.javapro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101JT Download the code for this article.

JP0101JT_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.


www.develop.com
LOS ANGELES • BOSTON • PORTLAND • LONDON
Schedule of training courses, and other developer resources are available by calling 800.699.1932 or on our web site.
"mind the gap"
Java • JDK • VM • JTS
XML • JNDI • EJB
Security • JSP • J2EE
JAAS • JDBC • JMS
RMI • JNI • JVMPI

TURN XML INTO EXCEL®

Build dynamic reports on Web and application servers with the Formula One for Java iReporting Engine

Formula One for Java 8.0 attaches to data sources on servers, performs calculations, and dynamically generates and distributes reports in richly formatted Excel spreadsheets, HTML, or images on the Web — all from an API-driven, Excel-compatible reporting engine that you embed in your servlets, EJB, JSP, applets, and Java applications.

- **Ideal platform for building Java-based reporting applications**

iReporting™ for eBusiness

- **Includes internal XSLT processor for filtering and formatting JDBC, XML, and tab delimited data into spreadsheet templates**
- **Dynamically generates Excel reports including formulas, worksheet functions, charts, merged cells, rotated text, outlining, and conditional formatting**
- **Also delivers reports to Web browsers as formatted HTML or through a custom applet that displays live cross-platform spreadsheets**
- **100% Pure Java with no reliance on Windows or Excel**

Attach to data sources.

Filter data into spreadsheet-based report templates. Apply formatting. Perform scalable calculations.

Dynamically generate and distribute reports in richly formatted Excel, HTML, or Web-ready images.

FORMULAONE™ iREPORTING ENGINE FOR JAVA

WWW.TIDESTONE.COM
800-884-8665

FREE 30-DAY TRIAL

Copyright © 2000 Tidestone Technologies, Inc. All rights reserved. Tidestone, iReporting, and Formula One are trademarks of Tidestone Technologies, Inc. Java, 100% Pure Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the U.S. and other countries. Excel is a trademark of Microsoft Corporation. All other trademarks are property of their respective owners. Specifications subject to change without notice.

devx marketplace

development exchange

Your #1 Source for

Software

Stack Overflow Mozquito Factory
Get dynamic pages without scripting XHTML
Format: Download
$145

Steema Software SL TeeChart Pro 4.0 VCL
Deliver high performance network solutions
Professional Delphi 5
Format: Download
$179

APEX True DBSuite 6.0
The complete solution for database front-end applications
Format: Download
$679
Upgrade for $559

eHelp RoboHelp Office 9.0
Provides a powerful native environment for authoring new content—works hand-in-hand with printed documentation
Format: Download
$899

Red-Gate SQL Compare
Compare databases structures of Microsoft SQL Server 7 and MSDE databases
Format: Download
$45

ProtoView JFCSuite 4.0
Lightweight components with heavyweight features
Format: Download
$899

Addtional versions and editions available online!

LGB Systems XControls
Quickly develop scalable and robust database applications
Enterprise Edition
Format: Download
$329

FarPoint List Pro
Include up to 2 billion list items in your control
Format: Download
$179

VideoSoft® VSFlexGrid Pro® 7
Gives VB developers a comprehensive feature set for display and presentation of dynamic data
Format: Download
$249

Desaware SpyWorks
Extend the capabilities of VB to rival those of C++
Standard Edition
Format: Download
$169

Pervasive Tango™ 2000
A powerful solution for your Web-based applications
Application Server Standard Edition for Windows NT
Format: CD ROM
$2,999

Sheridan™ Components Suite™
Combines the best of Sheridan's ActiveX components in one package
Format: Download
$319

netExs ASP+
Delivers a new paradigm for visual Web application development
Format: Download
$750

Order and Download Your Favorite Tools Now!

Choose the top development tools and components from these leading publishers:

We also carry products by:
Advantageware
CodeAvionics
Desaware
Sax Software
Stack Overflow
Steema Software

Order Online and Save!

http://software.devx.com

Or call 800.848.5523 or 650.833.7100 (outside the USA)

DevX.com MarketPlace prices are subject to change without notice. The Development Exchange, DevX, and DevX.com are trademarks of DevX.com, Inc. All products listed are registered marks or trademarks of their respective owners.

Development Tools!

Make the Transition Now to Microsoft® Windows® 2000

Microsoft® Windows® 2000 Professional
Deploy Windows 2000 on the desktop and realize significant performance advantages over previous releases
$275
Upgrade for $129 (after $70 mail-in rebate)

Microsoft® Windows® 2000 Server
The new standard in power and reliability

5 User License	$879
10 User License	$1,049
25 User License	$1,529
5 User License, Version Upgrade	$446
10 User License, Version Upgrade	$529
25 User License, Version Upgrade	$739

Microsoft® Windows® 2000 Advanced Server
Designed for demanding enterprise applications and database-intensive work

25 User License	$1,789

Addtional versions and editions available online!

New Releases!

Also check online for great deals on these other Microsoft® products:

GO

- Microsoft® Commerce Server 2000
- Microsoft® Host Integration Server 2000
- Microsoft® Internet Security and Acceleration Server 2000
- Microsoft® Application Center 2000
- Microsoft® SQL Server 2000 Standard Edition
- Microsoft® SQL Server 2000 Enterprise Edition
- Microsoft® BizTalk Server 2000
- Microsoft® Access 2000
- Microsoft® Outlook 2000
- Microsoft® Project 2000

Microsoft®

Books

Get up to speed with the best books to help you do your job better and faster.

Building Linux Clusters
Introduces you to the basics of cluster installation and configuration
By David HM Spector
$35.96

Scaling Oracle8i
Understand exactly what scalability means in the enterprise!
By James Morle
$44.95

Advanced Visual Basic 6 : Power Techniques for Everyday Programs
Learn how to write better VB code with new, advanced techniques.
By Matthew J. Curland
$31.96

Enterprise Application Integration with XML and Java
Includes techniques for building flexible, extensible EIA solutions
By JP Morgenthal
$35.99

Presenting C#
Get up to speed and begin creating next-generation applications and Web services today
By Christoph Wille
$20

A Preview of Active Server Pages+
Includes a supporting Web site with features, changes and updates to ASP+.
By Alex Homer, David Sussman, Richard Anderson & Robert Howard
$27.99

Debugging Visual Basic
Prevent pesky errors from slipping into your VB apps.
By David G. Jung, Jeff Kent
$31.99

Check out our complete book inventory covering:

.NET	COM/DCOM	Security
Access	Databases	SQL Server
Active Server Pages	Dynamic HTML	Visual Basic
ActiveX	Enterprise Development	Visual InterDev
ADO	Exchange and Outlook	Web Development
C++	Java	Windows 2000
Certification	Linux	XML
	MTS	

To Order:

Go online at http://software.devx.com

or http://books.devx.com

or, e-mail us at marketplace@devx.com

Please call us directly for personalized service on higher volume pricing and purchases, or for additional information.

OVERNIGHT EXPRESS DELIVERY

1- 800-848-5523
(outside the US call 1-650-833-7100)
(8:30am - 5:30pm M-F, Pacific Time

devX development exchange

http://books.devx.com

Or call 800.848.5523 or 650.833.7100 (outside the USA)

Visual Components

Swinging on a JCheckTree

The flexible JCheckTree component lets you create a JTree with selectable checkboxes

BY CLAUDE DUGUAY

Selecting elements from a JTree is one of those exercises that seems like it should be much simpler to do than it is. Swing provides incredible power in its architecture, but the last mile is often the most difficult to travel. Let's develop a reusable set of classes that facilitates the development of a JTree with checkbox selections. To maximize flexibility, the node, renderer, and editor objects will support the containment of other node elements and renderers so that existing behavior can be delegated easily.

Our implementation allows you to plug in your own renderers and data elements by wrapping a CheckTreeRenderer and CheckTreeEditor class around the renderer you want to use. To keep things simple, we use a default renderer class called DefaultTreeCellRenderer. For demonstration purposes, I'll provide a BasicTreeNode and associated BasicTreeCellRenderer to support arbitrary text nodes with alternate icons. As always, limited space precludes the listing of all the classes, so you'll find them all online at www.java-pro.com.

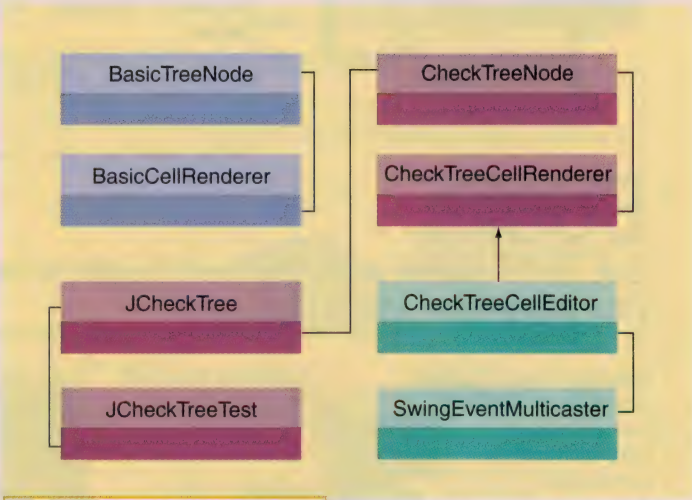


Figure 1 Check out the Classes These are the classes used in the CheckTree component.

A couple of comments about Swing are important at this point. First, Swing doesn't like to reference the same instance of a renderer in both the CheckTreeRenderer and CheckTreeEditor classes we'll use, so you'll have to create a second instance to avoid problems. I couldn't determine why this is such a big issue, but it is. If you don't create a second instance, you'll have no end of heartache, since what you might have done wrong won't be clear.

TreeCellEditor implementations also suffer from an interesting problem when used in a JTree component. The getCellEditorValue() method has to return a modified version of a mutable node to make this work properly. A clone or new instance of the modified node appears to cause problems when it comes to redrawing the tree. These two problems alone compelled me to do a fair amount of debugging when I implemented this strategy, but the rest of this implementation was relatively problem-free. This walkthrough should help you understand what it takes to implement this solution and leave you with a set of reusable classes that makes this whole process considerably easier.

The only significant shortcoming with our delegation approach is that alternate editors are not supported. The assumption is that you can use alternate renderers if you need them, but you'll need to use the editor to handle the checkbox selection. A more complicated and comprehensive solution is possible, but well beyond the scope of this discussion. In most cases, you will need to render custom nodes, but the need for custom editors will, most likely, be rare.

We'll throw a class into the mix that you might find interesting if you develop a lot of Swing components that need to implement a listener interface. SwingEventMulticaster inherits from the Java AWTEventMulticaster and implements the same list-based multicaster for all the Swing events and listeners. If you've never used the AWTEventMulticaster, it's well worth reading the JavaDoc description, which tells you how to apply the methods you'll need to use.

The principle behind the AWTEventMulticaster is simple enough and uses a linked list model to manage a thread-safe listener list. You can use the add() and remove() methods to add and remove listeners to and from the list, calling respective event methods directly or through delegation as you need them. This approach is very useful and easier to apply than the listenerList solution of-

fered in Swing. We don't have room to take a closer look, but you can find the code online at www.java-pro.com along with the rest of this project.

Take a look at Figure 1, which shows the classes used in the CheckTree component. The JCheckTree test (see Figure 2) uses the BasicTreeNode and BasicCellRenderer to demonstrate the use of alternate renderers. You can use the DefaultTreeNode and DefaultTreeCellRenderer to accomplish the same results, but I wanted to use a variant to make it clear that you can use any suitable node/renderer combination, so it was important to include these in the JCheckTreeTest environment. We won't say much about these classes, since they are not critical in this implementation.

Selecting elements from a JTree should be simpler to do than it is. This reusable set of classes facilitates the development of a JTree that makes this process simple

The four classes that deserve attention are CheckTreeNode, CheckTreeRenderer, CheckTreeEditor, and JCheckTree. Let's take a look at each of these in turn.

Listing 1 is the CheckTreeNode class. We extend DefaultMutableTreeNode and add the ability to set or get the current boolean selection state, while supporting any user-defined data element with setUserObject() and getUserObject(). The three constructors provide settings for the user object, the selection state, and a flag indicating whether we should propagate selections to the children of this node. This feature is useful when working with checkboxes, since the selection of a parent node often implies selecting its children. Naturally, we want this behavior to be based on circumstances, so we provide access to the flag directly in the constructor.

CheckTreeNode exposes a pair of accessor methods—isSelected() and setSelected()—to work with the selection state. The propagateSelected() method, called only when the propagate flag is set, uses the children() method from the TreeNode interface to enumerate through the node's children and reset the selection state on each child. You'll notice we override the setUserObject() method. For some reason, if you don't filter out the condition where the current node is being set, you can end up in a recursive loop that causes a stack overflow. This problem can be avoided easily by ignoring that condition and returning immediately.

The CheckTreeRenderer class (see Listing 2) allows us to show the current selection state in a JCheckBox, rendering the user object with a specified renderer. Both constructors expect a reference to the parent JTree and an optional renderer. By default, we use the DefaultTreeCellRenderer if one is not specified. The display component is

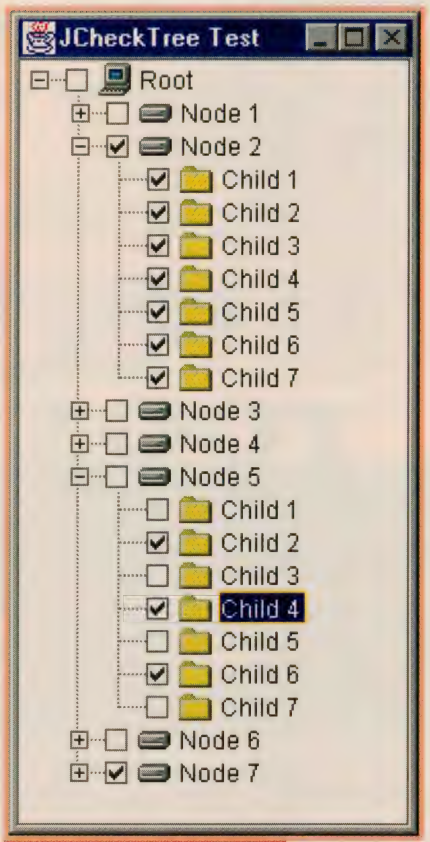


Figure 2 The Tree in Bloom In our JCheckTree application, high-level select and deselect operations are automatically propogated down child nodes.

Listing 1 Nodes on a Tree The CheckTreeNode class has three constructors that provide settings for the user object, the selection state, and a flag indicating whether we should propagate selections to the children of this node—a useful feature when working with checkboxes.

```
import java.util.*;
import javax.swing.tree.*;

public class CheckTreeNode
  extends DefaultMutableTreeNode
{
  protected boolean selected, propagate;

  public CheckTreeNode(Object data)
  {
    this(data, false, true);
  }

  public CheckTreeNode(Object data, boolean selected)
  {
    this(data, selected, true);
  }

  public CheckTreeNode(Object data,
    boolean selected, boolean propagate)
  {
    super(data);
    this.selected = selected;
    this.propagate = propagate;
  }

  public boolean isSelected()
  {
    return selected;
  }

  public void setSelected(boolean selected)
  {
    this.selected = selected;
    if (propagate)
      propagateSelected(selected);
  }

  public void propagateSelected(boolean selected)
  {
    Enumeration enum = children();
    while (enum.hasMoreElements())
    {
      CheckTreeNode node =
        (CheckTreeNode)enum.nextElement();
      node.setSelected(selected);
    }
  }

  public void setUserObject(Object obj)
  {
    if (obj == this) return;
    super.setUserObject(obj);
  }
}
```

Listing 2 Tree Rendering The CheckTreeRenderer class lets you show the current selection state in a JCheckBox, rendering the user object with a specified renderer.

```
import java.awt.*;
import java.util.*;
import javax.swing.*;
import javax.swing.tree.*;

public class CheckTreeCellRenderer extends JPanel
  implements TreeCellRenderer
{
  protected CheckTreeNode node;
  protected TreeCellRenderer renderer;
  protected JCheckBox check;

  public CheckTreeCellRenderer(JTree tree)
  {
    this(tree, new DefaultTreeCellRenderer());
  }

  public CheckTreeCellRenderer(
    JTree tree, TreeCellRenderer renderer)
  {
    setOpaque(false);
    setLayout(new BorderLayout());

    this.renderer = renderer;
    add(BorderLayout.CENTER,
      renderer.getTreeCellRendererComponent(
        tree, "", true, true, true, 0, true));

    check = new JCheckBox();
    check.setMargin(new Insets(0, 0, 0, 0));
    check.setBorderPaintedFlat(true);
    check.setOpaque(false);
    add(BorderLayout.WEST, check);
  }

  public Component getTreeCellRendererComponent(
    JTree tree, Object value, boolean selected,
    boolean expanded, boolean leaf, int row,
    boolean hasFocus)
  {
    if (value instanceof CheckTreeNode)
    {
      node = (CheckTreeNode)value;
      check.setSelected(node.isSelected());
      value = node.getUserObject();
    }
    renderer.getTreeCellRendererComponent(
      tree, value, selected, expanded, leaf, row, hasFocus);
    return this;
  }
}
```

Listing 3 Checking your Edits In the CheckTreeCellEditor class, the editor extends our CheckTreeCellRenderer and implements the TreeCellEditor and ActionListener interfaces.

```
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

public class CheckTreeCellEditor
  extends CheckTreeCellRenderer
  implements TreeCellEditor, ActionListener
{
  protected CellEditorListener list;

  public CheckTreeCellEditor(JTree tree)
  {
    this(tree, new DefaultTreeCellRenderer());
  }

  public CheckTreeCellEditor(
    JTree tree, TreeCellRenderer renderer)
  {
    super(tree, renderer);
    check.addActionListener(this);
  }

  public Component getTreeCellEditorComponent(
    JTree tree, Object value, boolean selected,
    boolean expanded, boolean leaf, int row)
  {
    return getTreeCellRendererComponent(
      tree, value, true, expanded, leaf, row, true);
  }

  public boolean stopCellEditing()
  {
    return true;
  }

  public Object getCellEditorValue()
  {
    node.setSelected(check.isSelected());
    return node;
  }

  public boolean isCellEditable(EventObject event)
  {
    return true;
  }

  public boolean shouldSelectCell(EventObject event)
  {
    return true;
  }

  public void  cancelCellEditing()
  {
    fireEditingCanceled();
  }

  public void addCellEditorListener(
    CellEditorListener listener)
  {
    list = SwingEventMulticaster.add(list, listener);
  }

  public void removeCellEditorListener(
    CellEditorListener listener)
  {
    list = SwingEventMulticaster.remove(list, listener);
  }

  protected void fireEditingStopped()
  {
    if (list != null)
      list.editingStopped(new ChangeEvent(this));
  }

  protected void fireEditingCanceled()
  {
    if (list != null)
      list.editingCanceled(new ChangeEvent(this));
  }

  public void actionPerformed(ActionEvent event)
  {
    fireEditingStopped();
  }
}
```

January 22-23, 2001
Hyatt Regency San Francisco, CA

THUNDER LIZARD PRODUCTIONS

PHOTOSHOP Conference 2001

FOCUS ON 6.0

PLUS SPECIAL FOCUS DAY
WEB FOCUS OR FINE ART FOCUS
January 24

A New Century. A New Photoshop.

By attending the Photoshop Conference you will learn from the top artists, teachers, and industry leaders—many of which have been working with Photoshop 6.0 for over a year. All sessions will delve into the most important changes in Photoshop 6.0. We've included sessions dedicated to the new Layer features, Vector Masking, powerful Typography controls, and the reworked Color Management engine.

Additionally we've scheduled innovative sessions on working with high-bit data, distorting reality with the new Liquify feature, and how to achieve museum quality fine art prints. It's a new century, and a new Photoshop conference. Join us in San Francisco—and learn how to take Photoshop 6.0 to the next level.

ATTENDEE COMMENTS

"Outstanding! The professionalism of this conference is second to none. This entire production is PURE EXCELLENCE! Thanks for the illumination. Now I can see!"

–Cornell Dunmore, Associate Editor/Desktop Publisher, Int'l Assn of Machinists & Aerospace

"This was a very well thought out conference that was put together with the attendee in mind. I am coming away with a wealth of knowledge that I will be putting to immediate use in my design. Thanks!"

–Jodi Lau, Publications, Revival Ministries International

COSPONSORED BY

Adobe

DIGITAL PUBLISHING DESIGN GRAPHICS

internet.com

Thunder Lizard Productions
Conferences with Content

FTP A DIVISION OF FAWCETTE TECHNICAL PUBLICATIONS

GET THE DETAILS

thunderlizard.com/info

800/221-3806 or 206/285-0305

Listing 4 Watch out for that Tree The JCheckTree class ties everything into a usable component, subclassing JTree and setting the CheckTreeCellRenderer() and CheckTreeCellEditor(), as well as making the tree editable.

```
import java.awt.*;
import javax.swing.*;
import javax.swing.tree.*;

public class JCheckTree extends JTree
{
  public JCheckTree(TreeNode root)
  {
    super(root);
    super.setCellRenderer(new CheckTreeCellRenderer(this));
    setCellEditor(new CheckTreeCellEditor(this));
    setEditable(true);
  }

  public void setCellRenderer(TreeCellRenderer renderer)
  {
    super.setCellRenderer(
      new CheckTreeCellRenderer(this, renderer));
  }

  public void setEditorRenderer(TreeCellRenderer renderer)
  {
    setCellEditor(new CheckTreeCellEditor(this, renderer));
  }
}
```

a JPanel with the JCheckBox in the WEST position. Notice that we use the JDK 1.3 method setBorderPaintedFlat() to render the checkbox flatly against the background, rather than sunken as it normally would be. We also set a zero margin and make JCheckBox transparent with setOpaque() set to false.

The render component is initially created in the constructor to avoid having to add the rendering component repeatedly to the panel. Each subsequent call to getTreeCellRendererComponent() sets the component values for rendering, but we expect the reference to be the same so we can avoid the overhead of adding the component to the container, removing the previous reference, and recalculating the layout positions. This methodology is not only more efficient, but avoids inconsistencies in later rendering.

You'll find the code for CheckTreeCellEditor in Listing 3. The editor extends our CheckTreeCellRenderer and implements the TreeCellEditor and ActionListener interfaces. We use the ActionListener to listen to changes in the JCheckBox from the parent class. Whenever a change occurs, we call fireEditingStopped() to make sure any cell editor listeners get the message. In the case of JTree, this allows the view to be updated whenever the user makes a selection. The ActionListener is registered in the constructor, which supports the same arguments as the parent class.

Because custom user objects and renderers are supported, flexibility is maintained while the amount of work required to implement the checkbox behavior is reduced

The TreeCellEditor interface requires the implementation of seven methods from the CellEditor interface and the getTreeCellEditorComponent(), which closely mimics the getTreeCellRendererComponent(). We use the SwingEventMulticaster class to implement the addCellEditorListener() and removeCellEditorListener() methods, as well as the fireEditingStopped() and fireEditingCanceled() methods. Notice how simple and straightforward this is with the SwingEventMulticaster.

The CellEditor methods isCellEditable() and shouldSelectCell() both return true. The reason that stopCellEditing() also returns true is not obvious unless you consider that we're done editing the moment the user makes a selection, so we can return to rendering immediately. I found the getCellEditorValue() implementation difficult since the implication is that the return value can be a new instance of the required data element. In fact, the model is not updated this way, so this strategy works only if you modify an existing object—in our case by calling setSelected() with the current checkbox state.

JCheckTree ties everything into a usable component (see Listing 4). All we're doing here is subclassing JTree and setting the CheckTreeCellRenderer() and CheckTreeCellEditor(), as well as making the tree editable with a call to setEditable(). We override the setCellRenderer() method and add a new setEditorRenderer() method. The first of these is intended to make the use of our CheckTreeCellRenderer() transparent. The second allows us to set the same renderer for the editor.

Remember that you need to use a separate instance of the renderer you want to apply in the editor to avoid Swing-rendering problems. This is a way of insuring success without constraining your ability to use different renderers. You can see all this in action in the JCheckTreeTest, which you'll find online with the rest of the code. JCheckTreeTest creates a hierarchy of CheckTreeNode objects and a JCheckTree, setting two instances of the BasicTreeCellRenderer() with the setCellRenderer() and setEditorRenderer() methods—putting the JCheckTree in a JScrollPane for demonstration purposes.

JCheckTree is designed to let you work with selectable nodes easily. Because custom user objects and renderers are supported, flexibility is maintained while the amount of work required to implement the checkbox behavior is reduced. This requirement has become common in modern applications, so I hope you'll find this solution as reusable as it was intended. The SwingEventMulticaster class I threw in is one of those classes that should be part of the Swing release, but was never implemented. I hope you find this code as useful as I did. ■

Claude Duguay is vice president of engineering at 360 Powered Corp. He has developed software for more than 20 years. He is passionate about Java, reads about it incessantly, and writes about it as often as possible. You can reach him at claude@360.com.

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101VC Download the code for this article.

JP0101VC_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

XML Solutions and Strategies:

Get them here first!

How to Choose an XML Ser
XML
MAGAZINE
.NET=XML
How XML Will Move Microsoft Beyond Windows
Apps to Web Services with SOAP
Serving XML with JavaServer Pages
Expert Voices: .NET's Wascha on XLANG, Hejlsberg on C#
Cocoon Separates Logic from Layout
Microsoft .net

FREE Subscription!

XML Magazine is the only independent publication devoted 100-percent to the latest information about using XML for enterprise-level challenges. Created by the same experts who gave you *VBPJ, Exchange & Outlook* and *Java Pro,* this is the one indespensible XML resource that you can't afford to be without. Published every other month, you'll get practical, hands-on articles, tips and enterprise-level solutions, including:

- Pragmatic techniques for developing with XML
- Cutting-edge analysis of industry trends
- Independent product reviews and new product announcements
- Columns by the industry experts who design the technology
- Complete source code examples

Sign up today for your FREE subscription. Complete the postage-paid card in this issue.

FTP FAWCETTE TECHNICAL PUBLICATIONS

209 Hamilton Avenue, Palo Alto, CA 94301-2500 • Phone: 650.833.7100 • Fax: 650.853.0230

©2000, FTP, Inc. XML is a trademark of MIT. Publisher reserves the right to determine qualifications for free subscriptions.

Java To Go

Priority Inversion Redux

The issues of synchronization and threads in multitasking environments are critical problems for operating real-time systems

BY RICK GREHAN

Priority inversion is a kind of multitasking ailment that I investigated some time ago ("Impure Thoughts," *Java Pro*, February 1999). I thought of it again as I read the *Real-Time Specification for Java*, which includes an extensive discussion of scheduling, synchronization, and threads. Priority inversion is a matter of intense scrutiny by real-time developers—not because they enjoy wallowing in the study of maladies, but because avoiding this problem is critical for the proper operation of real-time systems. Let's revisit priority inversion and look at some solutions. (Look for resource links in the sidebar "For More Information.")

Recap

To manifest, priority inversion requires the following ingredients: three threads (call them A, B, and C), each running at a different priority level (call them high, medium, and low). (In this example, the name I gave to each priority level indicates a thread's relative execution eligibility. In other words, if the dispatcher has a choice of a high- and a medium-priority thread, both ready to run, it will select the high-priority thread.) Furthermore, threads A and C share access to a synchronization object—we'll simply call it a "lock"—that is binary: Only one thread is allowed to acquire the lock at any given time. Presumably, the lock guards a shared resource. A thread must acquire the lock to access the resource; it releases the lock when access is completed.

Finally, priority inversion requires a preemptive multitasking environment. Hence, it is possible that when a thread acquires the lock, the multitasking kernel could suspend that thread before the lock is released.

Now, imagine that threads A, B, and C begin running. At some point, thread C acquires the lock. Thread C's time slice expires before the thread completes access to the controlled resource, and thread C is suspended while still owning the lock. Thread A runs, and in so doing attempts to acquire the lock. Of course, the attempt fails, so thread A is suspended. At this time, thread B is ready to run.

The following scenario ensues. Thread A cannot run because it is blocked, waiting for the lock. Thread C cannot run because thread B has a higher priority than thread C, and thread B isn't blocked. Consequently, thread B literally owns the machine. Priority inversion has occurred, in that thread B appears to have traded priorities with thread A. Unless thread B somehow yields to other threads, this situation could go on indefinitely.

The Mars Pathfinder was a victim of priority inversion. As I recall, in the situation with Pathfinder, a watchdog timer recognized that the blocked high-priority thread had been unable to run for an unacceptably long time, and the system was rebooted.

Solution I – Priority Inheritance

You can avoid priority inversion in at least two ways—at least, the *Real-Time Specification for Java* recognizes two. To illustrate them both, I'll use our example of three threads and a lock.

The first solution is the "priority inheritance" scheme, so-called because it has the effect of causing one thread—temporarily, at least—to inherit the priority of another thread. Priority inheritance involves the intervention of the operating system kernel; at a minimum, it involves that part of the kernel that manages synchronization objects (see Figure 1).

For More Information

The public release of the *Real-Time Specification for Java* is available for download from http://java.sun.com/aboutJava/communityprocess/first/jsr001. Foundational material for the specification can be found in an earlier work, *Requirements for Real-Time Extensions for the Java Platform*, which is available at www.nist.gov/itl/div897/ctg/real-time/intro.html). This earlier work is the product of the requirements group for Java real-time extensions, and it includes lengthy discussion of terminology used in the *Real-Time Specification*. You should read the requirements document before tackling the specification.

The Mars Pathfinder was a victim of priority inversion, you can still read about it—and its cure—at http://www.cs.cornell.edu/cs614- sp98/papers/pathfinder.html.

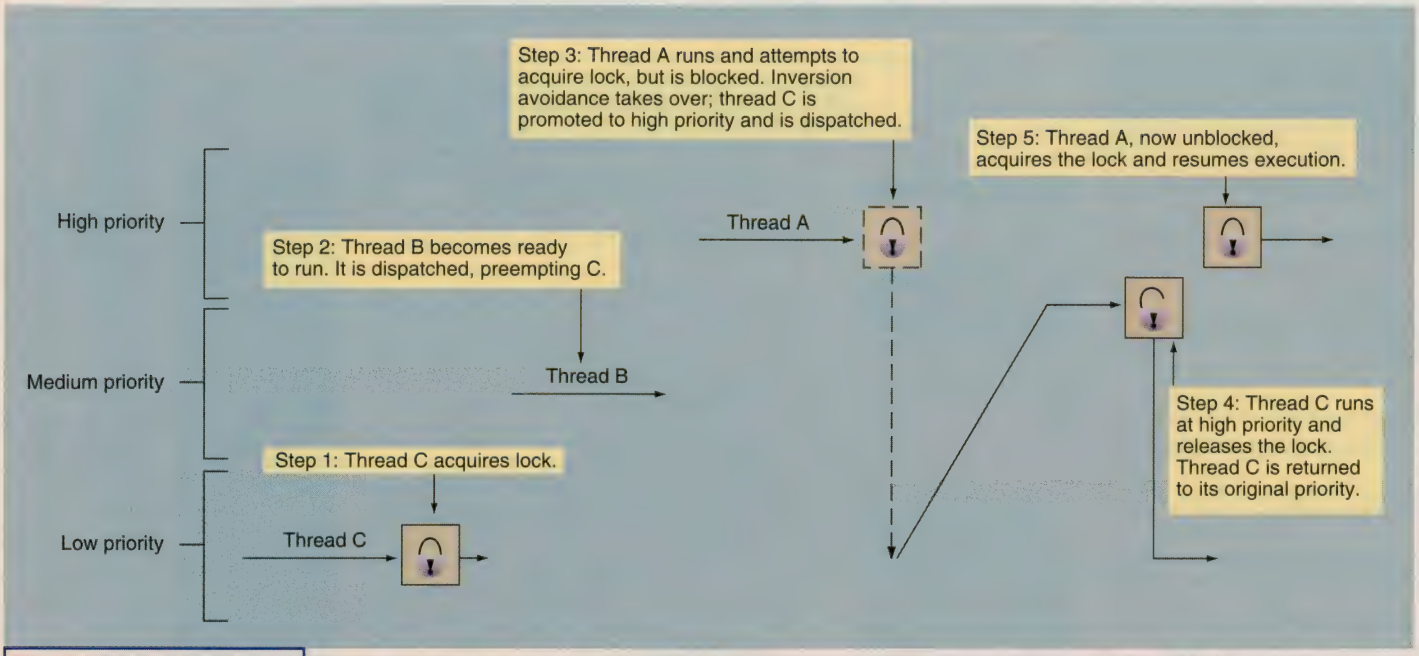

Step 3: Thread A runs and attempts to acquire lock, but is blocked. Inversion avoidance takes over; thread C is promoted to high priority and is dispatched.
Step 5: Thread A, now unblocked, acquires the lock and resumes execution.
High priority
Thread A
Step 2: Thread B becomes ready to run. It is dispatched, preempting C.
Medium priority
Thread B
Step 4: Thread C runs at high priority and releases the lock. Thread C is returned to its original priority.
Step 1: Thread C acquires lock.
Low priority
Thread C

Figure 1 Priority Inheritance Priority inheritance is one way of avoiding priority inversion. This scheme causes one thread to temporarily inherit the priority of another thread.

With priority inheritance in effect, when a thread attempts to acquire a lock and fails (because another thread owns the lock), code in the system examines the priority level of the requesting thread and compares that with the priority level of the thread owning the lock. If the priority of the lock owner is less than the priority of the lock requestor, the system temporarily promotes the priority of the owner to that of the requestor. So, the lock owner "inherits" the priority of the lock requestor.

With its priority promoted, the lock owner is no longer in peril of being blocked by threads whose priorities lay between its previous priority and its current priority. Nor will it be blocked by the lock requestor, because the two now possess equivalent priorities. Provided that no threads of still higher priority exist (even if they did, they would not threaten priority inversion), the lock owner would be scheduled to run, would execute, then release the lock.

One more bit of kernel-intervention occurs when the lock owner releases the lock. Specifically, the system returns the thread to its original priority level.

Had priority inheritance been employed, our three-thread scenario would have proceeded as follows. When thread A attempted to lock the lock object, the access request would have failed, and "anti-priority-inversion" code (somewhere in that part of the kernel that managed synchronization objects) would have gone into effect. That code would have seen that thread C's priority was lower than A's, and would have promoted C to high priority. So promoted, thread C would be scheduled to run ahead of B. Thread C would execute, access the shared resource, and ultimately release the lock. When the lock was released, another bit of "anti-priority-inversion" code would demote thread C back to flow priority. The lock thus released, thread A would unblock and be free to run.

This is the default priority inversion avoidance mechanism for real-time Java.

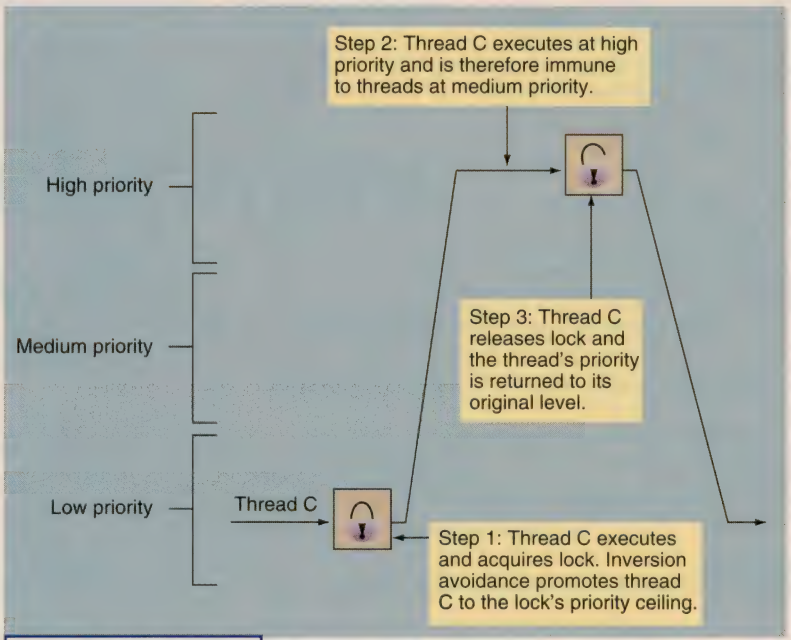

Step 2: Thread C executes at high priority and is therefore immune to threads at medium priority.
High priority
Step 3: Thread C releases lock and the thread's priority is returned to its original level.
Medium priority
Low priority
Thread C
Step 1: Thread C executes and acquires lock. Inversion avoidance promotes thread C to the lock's priority ceiling.

Figure 2 Priority Ceiling Priority ceiling is another way of avoiding priority inversion. This scheme involves the temporary promotion of priorities.

Solution II – Priority Ceiling

Another mechanism for priority inversion avoidance—also recognized by the *Real-Time Specification for Java*—is called "priority ceiling emulation." (The specification also refers to this as the "highest locker protocol.") As with priority inheritance, the priority ceiling scheme involves the temporary promotion of priorities. However, where priority inheritance works "behind the scenes" (that is, the programmer need not do anything explicit to enable priority inheritance), the priority ceiling scheme requires some up-front programming (see Figure 2).

Here's how it works. Any synchronization object used for mutual exclusion is given a priority ceiling. This is a parameter attached to that object at the time the object is created. This parameter is the "high-water mark" of priorities for threads wishing to acquire the synchronization object. In other words, only threads whose priority is at the ceiling or below may access the object.

Listing 1 Improved Elevator Program This listing shows a binary semaphore with priority ceiling capabilities.

```
import java.lang.Thread;
// Class BinarySemaphore
// This class has the same basic structure as the one
// appearing in the Feb. 99, JavaPro, with the difference
// that it implements priority ceilings.

class BinarySemaphore
{
  public BinarySemaphore(int ceil)
  {
    // Set ceiling
    ceiling = ceil;
    // Initially unlocked
    isLocked=false;
  }

  public void lock()
    throws CeilingViolationException
  {
    Thread thisThread;
    int tSavedPriority;

    // Get currently-executing thread
    thisThread = Thread.currentThread();

    // Check caller's priority
    if (thisThread.getPriority() > ceiling)
      throw new CeilingViolationException();

    // Priority OK, save current priority and promote the
    // thread
    // **COMMENT OUT NEXT 2 LINES TO
    // **TURN OFF P. I. AVOIDANCE
    tSavedPriority=thisThread.getPriority();
    thisThread.setPriority(ceiling);

    // Try to lock
    doLock();

    // Lock succeeded...REALLY
    // save priority
    // **COMMENT OUT NEXT LINE TO
    // **TURN OFF P. I. AVOIDANCE
    savedPriority = tSavedPriority;

  }

  private synchronized void doLock()
  {
    while (isLocked)
    { try
      { this.wait(); }
      catch (InterruptedException ex) { }
    }

    // Show locked
    isLocked=true;
  }

  public synchronized void unlock()
  {
    Thread thisThread;

    // Unlock
    isLocked=false;
    this.notify();

    // Demote the thread
    // **COMMENT OUT NEXT LINE TO
    // **TURN OFF P. I. AVOIDANCE
    thisThread = Thread.currentThread();
    thisThread.setPriority(savedPriority);
  }

  private boolean isLocked;
  private int ceiling;
  private int savedPriority;
}
```

CeilingViolationException.java

```
// Class CeilingViolationException
public class CeilingViolationException
  extends RuntimeException
{
  public CeilingViolationException()
  {
    super("Priority ceiling violation");
  }
}
```

myCanvas.java

```
import java.awt.*;
public class myCanvas extends Canvas
{
  public void paint (Graphics g)
  {
  }
}
```

Elevator.java

```
import java.awt.*;

class Elevator extends Thread
{
  public Elevator(myCanvas c, int place, int priority,
    Color col, BinarySemaphore bsema)
  {
    box = c;          // Set ref. to canvas
    x = place;        // Set loc. onscreen
    p = priority;     // Set priority
    clr = col;        // Set box color
    bs = bsema;       // Set ref. to semaphore
  }

  public void run()
  {

    // Dummy variable used for calculations to eat time
    double k = .5;

    // Set thread priority
    try { setPriority(p);
  }
  catch (IllegalArgumentException ex)
  { }

  // Draw first instance of the elevator
  Graphics g = box.getGraphics();
  g.setColor(clr);
  g.fillRect(x,y,XSIZE, YSIZE);
  g.dispose();

  // Run for a few thousand iterations
  for(int i = 1; i<=3000; i++)
  {
    move();

    // High and low-priority threads lock the semaphore
    if((x==10) || (x==90))
      try {
        bs.lock();
      }
      catch (CeilingViolationException ex)
      { }

      // All threads eat time calculating
      for (int j=1; j<=500000; j++)
        for (int l=1; l<=2; l++)
        k = Math.sin(k);
```

Continued on page 70.

Upgrade NOW

Premier Club Delivers:

- Savings on software
- Access to the DevX development article and code archive
- Live member-only events

Join the Premier Club for $3.95 per month and your first 30 days are free!

From the DevX home page click on JOIN

Need Development Know-How?

- Tap into 1000's of programming tips
- Ask the experts
- Browse tons of content and code from top IT publications
- Join live developer forums
- Access over 2,000 product reviews
- Purchase and download leading development tools
- **Join Premier Club** for exclusive benefits

Your search stops at DevX.

Get the unbiased information and tools you need NOW from the leading information resource for IT professionals developing business applications. We connect developers with objective, technical information and tools to get the job done—code samples, technical articles, discussions, learning services, and other resources aimed at helping you solve your toughest development challenges.

To learn more, visit:

www.devx.com

DevX is a trademark of DevX.com, Inc.

devx development exchange

(This is the up-front programming I mentioned.) Because you must explicitly set the synchronization object's priority ceiling when the object is created, you have to know the priorities of all the threads that might access that object beforehand. Of course, keeping track of the priorities of the threads in your application, and knowing which thread accesses which objects, is something you should be doing in any case. So, it's not like this requires unnecessary additional bookkeeping on the developer's part.

But the priority ceiling does something more. Any thread that acquires the synchronization object has its priority temporarily lifted to the object's priority ceiling. That guarantees that the object's owner will not be blocked by a thread of higher priority that also wants to acquire the object, nor will it be blocked by threads whose priority levels lie between the acquiring thread's original priority and the ceiling. When the thread releases the synchronization object, its priority is returned to its original level.

Continued from page 68.

```
      // High and low priority unlock
      if((x==10) || (x==90))
        bs.unlock();

      // All threads eat time calculating
      for (int j=1; j<=500000; j++)
        for (int l=1; l<=5; l++)
        k = Math.sin(k);

      // High priority sleeps some to give low priority
      // thread a chance to run (and cause inversion)
      if(x==10)
        try { Thread.sleep(800); }
        catch (InterruptedException ex) { }
    }
  }

  public void move()
  {
    Graphics g = box.getGraphics();
    Dimension d = box.getSize();

    g.setColor(clr);

    // Erase old
    g.clearRect(x,y,XSIZE,YSIZE);

    y+=yincr; // Advance elevator

    // If the elevator hits the top of the window,
    // reverse its direction
    if(y<0)
    {  y=0;
      yincr=-yincr;
    }

    // If the elevator hits the bottom of the window,
    // reverse its direction
    if(y+YSIZE>d.height)
    { y=d.height-YSIZE;
      yincr=-yincr;
    }

    // Draw the new rectangle
    g.fillRect(x,y,XSIZE,YSIZE);
    g.dispose();
  }

  private myCanvas box;
  private static final int XSIZE=10;
  private static final int YSIZE=10;
  private int x=0;      // Box location
  private int y=0;
  private int yincr=2;  // Move increment
  private int p;        // Priority
  private Color clr;    // Box color
  private BinarySemaphore bs;
}
```

ElevatorApp.java

```
import java.awt.*;
import java.lang.Thread;

// Class to implement Elevator application for
// illustrating priority inversion avoidance
public class ElevatorApp extends Frame
{
  public ElevatorApp()
  {
    setTitle("Elevator Thread Demo");
    c = new myCanvas();
    add("Center", c);
  }

  public void StartElevator()
  {

    // Create a new BinarySemaphore object with a ceiling of
    // Thread priority PRIORITY_HIGH
    bs = new BinarySemaphore(PRIORITY_HIGH);

    // Start first and third elevators
    e1 = new Elevator(c, 10, PRIORITY_HIGH, Color.black, bs);
    e1.start();

    e3 = new Elevator(c, 90, PRIORITY_LOW, Color.blue, bs);
    e3.start();

    // Let the two elevators run unmolested for a few seconds
    try {
      Thread.sleep(4000);
    }
    catch (InterruptedException ex)
    { }

    // Start 2nd elevator
    e2 = new Elevator(c, 50, PRIORITY_MED, Color.red, bs);
    e2.start();

  }

  public static void main(String[] args)
  {
    ElevatorApp f = new ElevatorApp();
    f.setSize(300,200);
    f.show();

    // Start the elevators
    f.StartElevator();

    // Wait for a while, then end
    try {
      Thread.sleep(70000);
    }
    catch (InterruptedException ex)
    { }

    System.exit(0);
  }

  private myCanvas c;
  private Elevator e1, e2, e3;
  private BinarySemaphore bs;
  private final static int PRIORITY_HIGH =
    Thread.NORM_PRIORITY + 2;
  private final static int PRIORITY_MED =
    Thread.NORM_PRIORITY;
  private final static int PRIORITY_LOW =
    Thread.NORM_PRIORITY - 2;
}
```

IT Nirvana.

When you feel a heightened state of being filthy rich.

Live your IT Nirvana.
Find it at dice.com.

Dice.com makes your IT Nirvana come true because we focus exclusively on quality high tech jobs with only the best companies. And we offer great career tools completely free of charge. So, we make it easy to find the perfect IT job, the biggest paycheck and, if you're lucky, spiritual enlightenment. What's your IT Nirvana?

A SERVICE OF:
EARTHWEB CAREER SOLUTIONS
Nasdaq: EWBX

dice.com
High tech jobs online

CareerLink
For IT Development Professionals

So, if the priority ceiling mechanism was used in the previous example, we would create the lock object with a priority ceiling of "high" (since thread A has the highest priority of any thread that would attempt to acquire the lock). When thread C acquired the lock, it would have its priority raised to high. Thread C is thereby guaranteed to run "ahead" of thread B and not be blocked. Thread C would still block thread A, but only as long as C owned the lock. When thread C released the lock, its priority level would be returned to low.

An Elevator Example

In that original *Java Pro* column, I illustrated priority inversion using a simple program that simulated three "elevators" moving up and down the screen. Each elevator was controlled by a thread, and each thread corresponded to one of the threads (A, B, and C) I used in this example. The elevators corresponding to threads A and C shared access to a binary semaphore (that is, lock) object. Those who ran the program saw (I hope) that that middle elevator ultimately gained control of the system.

I stated in that article that the key to avoiding priority inversion was strategically placed sleep(), yield(), or blocking-I/O calls. That's not entirely true. In fact, such measures are likely to relieve a priority-inverted situation only temporarily (as much as rebooting the Pathfinder solved its priority inversion problem only temporarily). For example, were I to take my own advice with the original "elevators" program, I would insert a yield() statement in the loop of the thread at medium priority. That would have solved the priority inversion problem—but only in spurts (which is hardly a solution).

In a real-time application, peppering threads with statements that block or yield is no solution. Some real-time applications consist of hundreds, perhaps thousands, of threads. In such situations, even with yield() statements sprinkled throughout medium-priority threads, it is unlikely that all medium-priority threads will yield simultaneously to allow the low-priority thread to run, release the lock, and permit the high-priority thread to run.

Rather than hope that filling threads up with yield statements will do the trick, a better solution is to simply implement a priority-inheritance avoidance mechanism and be done with it. This is easily done with the original elevator application, and Listing 1 shows the new application with a priority ceiling mechanism in place.

The class with the most changes is BinarySemaphore. The constructor takes an integer argument that is the priority ceiling. This value is saved in a private member. Also, I have divided the lock() method into unsynchronized and synchronized components. Basically, the synchronized portion, doLock(), is in an unsynchronized wrapper, lock(). The unsynchronized wrapper bumps the priority of the calling thread before entering the synchronized section. It has to be arranged that way, otherwise inversion might occur while the calling thread is inside doLock().

Once the thread successfully acquires the lock, its original priority is saved in a private integer, savedPriority. So, when the thread releases the lock by calling unlock(), the semaphore returns the thread to its original priority.

I've tried to tune the application to provide the right mix of thread workloads to exhibit inversion. I have also identified those lines in the BinarySemaphore listing that you can comment out to "turn off" the semaphore's priority ceiling capability. Depending on the platform on which you run the application, you might have to alter some of the loops and calls to sleep() to see inversion happen.

CareerLink
For IT Development Professionals

Recruit Professional Programmers
Through the Leading IT Development Magazines

CareerLink
For IT Development Professionals

Contact us today for more information about using the FTP Advertising Network and CareerLink to target half-a-million experienced, professional programmers, developers, IT managers, consultants and more.

To place an ad in this CareerLink section, please contact:

Kathy Bazan
Western Regional Manager
Phone: (650) 833-7148
Fax: (650) 853-0230
E-mail: kbazan@fawcette.com

Jean Anne Mahler
Eastern Regional Manager
Phone: (650) 833-7109
Fax: (650) 853-0230
E-mail: jamahler@fawcette.com

Run the application, then comment out the indicated lines, and rerun the application. On my 500 MHz NT system, the first run exhibited the high- and low-priority elevators trading off time almost equally (because of the sleeping done by the high-priority elevator). When the medium-priority elevator appeared, it and the high-priority elevator traded CPU time, because the medium-priority elevator does no sleeping and therefore never gives the low-priority elevator any time.

Occasionally, a system thread apparently blocked the medium-priority thread, and the low-priority thread got some execution time. When that occurred, the high-priority thread was temporarily stopped because the medium- and low-priority threads traded CPU time. (The low-priority thread had been promoted upon lock acquision, so it really wasn't a low-priority thread anymore.) When the low-priority thread exited the semaphore, it was demoted, and the situation returned to normal.

When I reran the application with the priority ceiling code commented out, the moment the low-priority thread was given some time, priority inversion set in, and only the medium-priority thread executed. This continued for some time; again, until something blocked the medium-priority thread, giving the low-priority thread a chance to run and unblock the high-priority thread.

Solutions for Real-Time Java

Java is a hybrid in that it did not begin life in a real-time environment (nor has it evolved in one). This creates some problems with thread synchronization for real-time Java. For example, *Real-Time Specification for Java* provides for a kind of real-time thread called NoHeapRealtimeThread, which does not create, delete, or reference objects on the heap. (It can reference objects in other memory regions that are not under the influence of the garbage collector. But I'll have to describe those in a later column.)

In effect, a NoHeapRealtimeThread is designed to run uninterrupted by the garbage collector (in fact, a NoHeapRealtimeThread can interrupt the garbage collector before a collection cycle is complete.) Hence, a NoHeapRealtimeThread runs at a priority level higher than the garbage collector. However, a normal Java thread cannot be raised to a priority level above that of the garbage collector. That means that there is no known priority inversion avoidance algorithm that can be devised ("correctly implemented" in the words of the *Real-Time* Java specification) that can handle a situation that involves a synchronization object shared between a NoHeapRealtimeThread and a normal Java thread.

The solution offered by the *Real-Time Specification for Java* is a set of wait-free queue classes. These provide protected, nonblocking, shared access to resources accessed by both regular threads and NoHeapRealtimeThreads. In function, they are not unlike the message queues that were used in the INMOS Transputer as the fundamental mechanism for synchronization. They are also a subject for a future column. ■

Rick Grehan is a technical evangelist with Compuware/NuMega. Rick was a senior editor at *BYTE* magazine (where he wrote the JavaTalk column) and at *Computer Design* magazine. He can be reached at rgrehan@top.monad.net.

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro.*

JP0101JG Download the code for this article.

JP0101JG_T Read this feature online. DevX Premier Club registration is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

JAVAPro

Subscribe Now!

Java Pro is the progessional developer's independent source for information about building serious Java applications.

Java Pro takes you beyond the hype by delivering hands-on, code-intensive, product-oriented information that helps you harness Java's unique capabilities in your enterprise-wide development efforts. Each month, Java Pro tackles tough development issues and provides unbiased, no-nonsense coverage of the latest tools on the market.

Call today to subscribe to Java Pro. You'll pay the low rate of $29.95 for a full year-thirteen issues in all. That's a savings of more than 61% off the newsstand rate. To subscribe now call 800-848-5523. Outside the U.S. call 650-833-7100. Or, order online at www.java-pro.com.

Fawcette Technical Publications
209 Hamilton Ave.
Palo Alto, CA 94301

Java is a registered trademark of Sun Microsystems, Inc.

http://www.scientific.com

Professional Software Developers
Looking for career opportunities?
* Java, J2EE, OOA/D
* Enterprise & n-tier Web-based Systems
* Distributed Syst. & Component Technologies
* Clients are Commercial Software Vendors, both pre-IPO and blue chips
Nationwide Service / Never a fee

Scientific Placement, Inc.
Toll-free 888-367-8251
mef@scientific.com

CareerLink
For IT Development Professionals

Pro Shop

Parsing XML for Beginners

Java isn't the last word on cross-platform computing—programs need to exchange data independently of their native platforms. Enter XML

BY DANIEL F. SAVARESE

Cross-platform computing is here to stay. In the early 1990s, many people thought the industry would consolidate around a single platform, and they heralded the demise of many an operating system (and, yes, a few did go by the wayside). But the rise of the Web and commercialization of the Internet, followed by the corporate acceptance of open source software, has helped a rich variety of platforms thrive. Although Java has played an important role in the multiplatform landscape, "write once, run anywhere" isn't the final word. On and off the Internet, programs need to communicate and exchange data independently of their native platforms. It is not sufficient for all programs to be written in Java to achieve interoperability.

We often directly correlate platforms with operating systems, leading to the erroneous conclusion that if we standardize on a single operating system, we will avoid all cross-platform integration issues. Any given operating system has many software offerings, and these programs need to exchange data with each other, either through files or by communicating over a network. Programs have to share everything from calendars, customer account information, and purchase orders to remote procedure calls, program data structures, and network protocol commands.

It used to be that if programs wanted to share documents with other similar programs (for example, one word processor reading a document created by another word processor), they needed to support the proprietary binary file formats of all the other programs. Creating searchable document libraries was difficult because most file formats made no distinction between content and presentation. This motivated the development of markup languages such as SGML, which described the structure of a document without imposing a specific interpretation of that structure. When the Web was conceived, the idea of a markup language was essential, given the need to display information on a wide variety of platforms. However, it was recognized that HTML had some cumbersome limitations, and an attempt was made to regularize its structure and make it more flexible, leading to the work on XML, the Extensible Markup Language.

Document publishing motivated the development of XML, but while it was being conceived, a strange thing happened. The Internet became the home of multitudes of network applications, many of which needed to speak to each other but could not do so easily. The value of XML for interapplication communication was quickly recognized, and a cross-platform revolution eclipsing the one spearheaded by Java was started. As many have observed, the role of XML in developing cross-platform Internet applications complements that of Java. Java gives you portable program logic, while XML gives you portable program data.

XML is not only well suited for sharing data between programs, it is an ideal format for configuration files. It is also becoming intertwined with Java standards. Servlets are configured with web.xml files, Java WebStart uses XML-based application deployment descriptors, and JSP pages support an XML representation. However,

Listing 1 Time Before Unix If XML had been invented before Unix, this is what /etc/passwd might have looked like.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<accounts>

  <account>
    <login>root</login>
    <passwd encryption="none">beer</passwd>
    <uid>0</uid>
    <gid>0</gid>
    <name>root</name>
    <home>/root</home>
    <shell>/bin/bash</shell>
  </account>

  <account>
    <login>jack</login>
    <passwd encryption="none">nobel</passwd>
    <uid>0</uid>
    <gid>0</gid>
    <name>Jack Kilby</name>
    <home>/home/jack</home>
    <shell>/bin/bash</shell>
  </account>

</accounts>
```

Listing 2 XML to Java XML is great, but sometimes you need to convert it into a Java object to do anything useful with it.

```
package example;

import java.util.*;

import org.w3c.dom.*;

public class Account {
  public static final int ENCRYPTION_NONE = 0;
  public static final int ENCRYPTION_MD5  = 1;
  public static final int ENCRYPTION_DES  = 2;

  private String __login;
  private String __password;
  private int __encryption, __uid, __gid;
  String __name, __home, __shell;

  public Account() {
    __login = __password = __name = __home = __shell =
      null;
    __encryption = ENCRYPTION_NONE;
    __uid = __gid = 0;
  }

  public void setLogin(String login) { __login = login; }

  public String getLogin()           { return __login; }

  public void setPassword(int encryption, String password) {
    __encryption = encryption;
    __password   = password;
  }

  public String getPassword()        { return __password; }

  public int getEncryption()         { return __encryption; }

  public void setUID(int uid)        { __uid = uid; }

  public int getUID()                { return __uid; }

  public void setGID(int gid)        { __gid = gid; }

  public int getGID()                { return __gid; }

  public void setName(String name)   { __name = name; }

  public String getName()            { return __name; }

  public void setHome(String home)   { __home = home; }

  public String getHome()            { return __home; }

  public void setShell(String shell) { __shell = shell; }

  public String getShell()           { return __shell; }

  /**
    * Returns a Unix style /etc/passwd entry.
    */
  public String toString() {
```

Continued on page 76.

don't throw out your Java properties files just yet. XML can be overkill if you have a small amount of configuration information. Remember, when you commit to XML, you commit to using an XML parser. This may be an unacceptable overhead for a small application or one that requires limited configuration support.

Should you decide to use XML for data interchange, network communication, or application configuration, an ample selection of parsers is at your disposal. One of the most popular is Xerces, an XML parser produced by the Apache XML Project (http://xml.apache.org). Xerces is true to the cross-platform nature of XML and provides a general API with Java, C++, and Perl bindings. If you have not yet delved into XML, think of it as markup where every open tag has a close tag. That simple characteristic has major implications—representing hierarchical information structures and parsing their representation is easy.

Listing 1 shows an example of what a Unix /etc/passwd file might have looked like had XML been around when Unix was developed. The first tag indicates the version of XML the document complies with and the text-encoding scheme. Parsers use this information to process the document properly. The next tag, <accounts>, is the root node of the document (every XML document must possess a root node). Following the root node, each account is listed, containing elements defining the login, password, user id, and other information typically found in /etc/passwd.

To do something useful with XML data inside a Java program, you must parse the XML and convert it to a more palatable form, usually a Java object. Listing 2 shows an Account class with getter and setter methods, as well as static methods to convert an XML document into an Account or Collection of Accounts. If an Account could not be created from the XML, an AccountParseException is thrown. AccountParseException simply subclasses Exception and has no special features:

```
package example;
public class AccountParseException
 extends Exception {}
```

Before the static methods can be used, the XML file must be parsed and stored in memory as a Document Object Model (DOM) Document. DOM is a platform-neutral interface to XML documents defined by the World Wide Web Consortium (W3C) that has multiple language bindings. (You can download the specification from www.w3.org/DOM.) The W3C provides a set of Java interfaces for manipulating documents in the org.w3c.dom package and its subpackages. Although these interfaces define how to manipulate XML documents, they do not define how to turn an XML file into an org.w3c.dom.Document instance.

We often directly correlate platforms with operating systems, concluding erroneously that if we standardize on a single operating system, we will avoid all cross-platform integration issues

Xerces provides a DOMParser class in its org.apache.xerces.parsers package, which can be used to construct a DOM Document from existing XML data. The Document is created from an InputSource, a class that abstracts the source of the XML. InputSource is part of the org.xml.sax package. SAX, the Simple API for XML, was an independently developed standard for event-driven parsing of XML, and is now a de facto standard. Event-driven parsing often can be more efficient and consume less memory than using a DOM representation,

because the entire document does not need to be stored in memory. However, the DOM structure is easy to manipulate inside a program, which is why I use it in the example programs.

A problem with directly creating a DOMParser is that we would really like to have plug-and-play parsers. If you use a parser implementation specific to a particular library, you probably will have to change a lot of code if you decide to switch to a different parser. SAX supports plug-and-play parsers with the XMLReader interface and XMLReaderFactory class. The recently finalized Java API for XML Parsing (JAXP) supports pluggable DOM parsers through DocumentBuilder and DocumentBuilderFactory abstract classes, and SAX parsers through SAXParser and SAXParserFactory. Xerces supports these classes, which are used in Listings 3 and 4, to create DocumentBuilder instances. Whether you use JAXP parser factories or parser factories of your own construction, writing your code to be parser independent is a good idea.

Parsing an XML file is straightforward because Xerces does all the work for you. First create an InputSource from a Reader, InputStream, or URL (see Listing 3). Then create a DocumentBuilder instance and call its parse() method with the InputSource as an argument. The parse() method returns a document reference to the parsed DOM structure. You have a lot of options for navigating a Document and turning it into a program data structure. If you understand the structure of your document well, querying the document by calling getElementsByTagName()—as is done in parseAccounts() in Listing 2—is convenient. As indicated by its name, this method returns a list of all the elements with a given tag name. Elements are represented by the Node interface; and a list of nodes by the NodeList interface. These are members of the org.w3c.org package for which Xerces provides its own implementations. If you swapped out Xerces for another DOM-compliant library, you would not have to make any changes to your code.

Once you have a reference to a Node, you can iterate through its children by getting a reference to its first child with getFirstChild(), and fetching each subsequent sibling with getNextSibling(), as is done in parseAccount() in Listing 2. You can reference the name of a Node with getNodeName(), its value with getNodeValue(), and its attributes with getAttributes(). The parseAccount() method uses the results of these methods to decide how to construct an Account object. For example, if a Node's name is "login," its value is used to set the Account login. The attributes of a Node are a little bit awk-

Continued from page 75.

```
    StringBuffer buffer = new StringBuffer();

    buffer.append(__login);
    buffer.append(":");
    buffer.append(__password);
    buffer.append(":");
    buffer.append(__uid);
    buffer.append(":");
    buffer.append(__gid);
    buffer.append(":");
    buffer.append(__name);
    buffer.append(":");
    buffer.append(__home);
    buffer.append(":");
    buffer.append(__shell);

    return buffer.toString();
  }

  public static final Account parseAccount(Node node)
    throws AccountParseException
  {
    Account account = new Account();

    node = node.getFirstChild();

    try {
      while(node != null) {
        String name = node.getNodeName();
        int id;

        if("login".equals(name)) {
          account.setLogin(
            node.getFirstChild().getNodeValue());
        }else if("passwd".equals(name)) {
          NamedNodeMap attributes;
          String password;

          password = node.getFirstChild().getNodeValue();
          attributes = node.getAttributes();

          name =
            attributes.getNamedItem(
            "encryption").getNodeValue();

          if("none".equals(name))
            id = ENCRYPTION_NONE;
          else if("md5".equals(name))
            id = ENCRYPTION_MD5;
          else if("des".equals(name))
            id = ENCRYPTION_DES;
          else
            throw new AccountParseException();

          account.setPassword(id, password);
        }else if("uid".equals(name)) {
          name = node.getFirstChild().getNodeValue();
          id = Integer.parseInt(name);
          account.setUID(id);
        }else if("gid".equals(name)) {
          name = node.getFirstChild().getNodeValue();
          id = Integer.parseInt(name);
          account.setGID(id);
        }else if("name".equals(name)) {
          account.setName(
            node.getFirstChild().getNodeValue());
        }else if("home".equals(name)) {
          account.setHome(
            node.getFirstChild().getNodeValue());
        }else if("shell".equals(name)) {
          account.setShell(
            node.getFirstChild().getNodeValue());
        }

        node = node.getNextSibling();
      }
    } catch(NumberFormatException nfe) {
      throw new AccountParseException();
    }
    return account;
  }

  public static final void parseAccounts(
    Document document, Collection accounts)
    throws AccountParseException
  {
    int length;
    NodeList list;

    list = document.getElementsByTagName("account");
    length = list.getLength();

    for(int i=0; i < length; i++) {
      Node node = list.item(i);
      Account account = parseAccount(node);
      accounts.add(account);
    }
  }
}
```

http://careerlink.devx.com

Upgrade Your Career

Connect with today's top programming jobs and talent on the premier online resource for IT professionals.

Find your next job

- Post a confidential resume
- Search 1000's of IT job listings

Build your team

- Post contract positions
- Find contractors
- Get highly-qualified resumes
- Post job listings

Find a contract gig

- Search gigs
- Post your profile

Whether you're looking for a great job or hard-to-find software development talent in a hurry, get the help you need at DevX.com's new CareerLink Zone. We've brought together the best resources on the Web to help IT development professionals, corporate recruiters, and hiring managers find qualified jobs and candidates fast.

Plus, CareerLink's **FREE** resources now include specialized career tips and advice for development professionals including:

CareerLink Newsletter—free development career news, tips and articles.

Salary Survey—salaries are higher than ever, see how yours compares.

devx development exchange

DevX.com, Inc.
913 Emerson, Palo Alto, CA 94301
www.devx.com

Phone: 650.566.2000

CareerLink http://careerlink.devx.com

POWERED BY

dice.com High tech jobs online

career central for developers

guru.com

DevX and The Development Exchange are trademarks of DevX.com, Inc. All other trademarks are the property of their respective owners.

Listing 3 Converting Passwords XML was invented after Unix, so this example program converts a passwd.xml file into /etc/passwd format.

```
package example;

import java.util.*;
import java.io.*;
import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.*;

public class ConvertToPasswd {

  // Converts XML data to a Unix style passwd file and
  // prints to stdout.
  public static final void main(String args[]) {
    InputSource input;
    Document document;
    ArrayList list;
    Iterator iterator;
    DocumentBuilderFactory factory;
    DocumentBuilder builder;

    try {

    }catch(Exception e) {
      e.printStackTrace();
      return;
    }

    if(args.length != 1) {
      System.err.println("usage: ConvertToPasswd xmlfile");
      return;
    }

    try {
      factory  = DocumentBuilderFactory.newInstance();
      input    = new InputSource(new FileReader(args[0]));
      builder = factory.newDocumentBuilder();
      document = builder.parse(input);
      list     = new ArrayList();

      Account.parseAccounts(document, list);

      iterator = list.iterator();

      while(iterator.hasNext()) {
        Account account = (Account)iterator.next();
        System.out.println(account.toString());
      }
    }catch(IOException ioe) {
      ioe.printStackTrace();
      return;
    }catch(SAXException saxe) {
      saxe.printStackTrace();
      return;
    }catch(AccountParseException ape) {
      ape.printStackTrace();
      return;
    }catch(ParserConfigurationException pce) {
      pce.printStackTrace();
      return;
    }
  }
}
```

Listing 4 Who Goes There? This example demonstrates how to authenticate a login against a passwd.xml file, but omits encryption handling.

```
package example;

import java.util.*;
import java.io.*;
import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.*;

public class Authenticate {

  public static final void main(String args[]) {
    InputSource input;
    DocumentBuilderFactory factory;
    DocumentBuilder builder;
    Document document;
    ArrayList list;
    Iterator iterator;
    String login, password;

    if(args.length != 3) {
      System.err.println(
        "usage: Authenticate xmlfile login password");
      return;
    }

    login    = args[1];
    password = args[2];

    try {
      factory  = DocumentBuilderFactory.newInstance();
      input    = new InputSource(new FileReader(args[0]));
      builder  = factory.newDocumentBuilder();
      document = builder.parse(input);
      list     = new ArrayList();

      Account.parseAccounts(document, list);

      iterator = list.iterator();

      while(iterator.hasNext()) {
        Account account = (Account)iterator.next();
        if(login.equals(account.getLogin())) {

          if(account.getEncryption() ==
            Account.ENCRYPTION_NONE &&
            password.equals(account.getPassword())) {
            System.out.println("Login authenticated.");
            return;
          }else
            break;
        }
      }

      System.out.println("Login authentication failed.");
    } catch(IOException ioe) {
      ioe.printStackTrace();
      return;
    } catch(SAXException saxe) {
      saxe.printStackTrace();
      return;
    } catch(AccountParseException ape) {
      ape.printStackTrace();
      return;
    } catch(ParserConfigurationException pce) {
      pce.printStackTrace();
      return;
    }
  }
}
```

ward to work with. They are stored in a NamedNodeMap, which is basically a hashtable containing attributes (also represented as Nodes—every part of a document is a Node). In Listing 1, the "passwd" Node has an "encryption" attribute that is used to decide how to interpret the password. The Authenticate program in Listing 4 doesn't use this attribute, but it is included for demonstration.

Before, if programs wanted to share documents with other similar programs, they needed to support the proprietary binary file formats of all the other programs. Now XML provides portable program data

As you can see, parsing an XML document can be simple, yet cumbersome. If you intend to convert XML data to JavaBeans, tools are now available for doing that, so you might want to avoid writing code such as that found in the Account class. However, XML can be applied broadly in Java applications. It makes an excellent vehicle for defining network protocol commands, although the overhead of parsing it may make it inappropriate for re-implementing all of the IETF protocols based on human-readable text commands such as SMTP, HTTP, and NNTP. Nonetheless, standards such as Simple Object Access Protocol (SOAP) may well establish XML as the underlying data representation format for distributed computing. Few would have predicted such a possibility when XML and Java were just getting started. ■

Daniel F. Savarese is a developer of handheld and wireless applications. He has worked as a senior scientist at the California Institute of Technology and director of Java product development at ORO, Inc. Daniel has written several Java packages, including the OROMatcher and PerlTools regular expression libraries and the NetComponents network protocol library. He is also a co-author of *How to Build a Beowulf*. Daniel can be reached at dfs@savarese.org.

Go Online

Use the following DevX Locator+ codes at www.java-pro.com to go directly to these related resources.

JP0101 Download all of the code for this issue of *Java Pro*.

JP0101PS Download the code for this article.

JP0101PS_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

Advertiser Index

Appstream 866-AppStream www.appstream.com	29	
Compuware Corporation 800-4-NUMEGA www.compuware.com/numega	39	
Developmentor 800-699-1932 www.develop.com	56	
DevX.com 888-963-3389 www.devx.com	69	
DevX CareerLink 888-963-3389 careerlink.devx.com	77	
DevX MarketPlace 888-963-3389 marketplace.devx.com	58, 59	
eHelp Corporation 800-563-0590 www.ehelp.com	31	
Exchange & Outlook Multimedia Resource 800-848-5523 exchange.devx.com	14	
Fiorano www.fiorano.com	8	
InformIT www.informit.com	37	
Inprise Corporation www.borland.com	43	
IONA Technologies 800-672-4948 www.iona.com	6, 7	
Java Pro Magazine Subscriptions 800-848-5523 www.java-pro.com	33	
Sitraka Software 888-361-3264 www.sitraka.com/greatapp	1	
Sitraka Software 888-361-3264 www.sitraka.com/deploy	C3	
Sitraka Software 888-361-3264 www.sitraka.com	C4	
LOOX Software 800-684-5669 www.loox.com	47	
Metrowerks, Inc. 800-377-5416 www.metrowerks.com	53	
POET Software 800-950-8845 www.poet.com	13	
Programmer's Paradise 800-445-7899 www.programmersparadise.com	15	
Progress Software 800-999-4599 www.sonicmq.com	17	
ProtoView Development Corp. 800-231-8588 www.protoview.com	2, 3	
RadView 888-RadView www.radview.com	23	
Rational Software www.rational.com	25	
RSA Security www.rsasecurity.com	19	
SilverStream 888-823-9700 www.silverstream.com/onlineseminars	10	
ThunderLizard Conferences 800-221-3806 thunderlizard.com/data	63	
Tidestone Technologies 800-884-8665 www.tidestone.com	57	
Togethersoft Corporation 919-833-5550 www.togethersoft.com/jp/	C2	
Unify Corporation 800-GO-UNIFY www.UnifyEWave.com	27	
Visual C++ Developer's Journal Magazine Subscriptions 800-848-5523 www.vcdj.com	49	
VisualSoft 800-516-0101 www.programmersparadise.com/visualsoft	15	
WebGain 888-822-3409 www.webgain.com	50	
XML Subscriptions 800-848-5523 www.xmlmag.com	65	
Zero G 415-512-7771 www.ZeroG.com	4	

Showcase

ID Automation.com www.IDautomation.com	46

CareerLink

Scientific Placement 888-367-8251 www.scientific.com	73
CareerLink Network	71, 72

Product Information

- To get immediate details on products advertised in this issue, you can simply log onto our online services at **http://infoasap.devx.com**.
- This provides a list of direct links to this month's advertiser's Web sites.

Advertising Sales Contacts

- **Shana Deane, Advertising Director** 413.665.6910, sdeane@fawcette.com
- **Beatrice Stonebanks, Southern Regional Sales Manager** 650.833.7163, bstonebanks@fawcette.com
- **David Lyman, Eastern Regional Sales Manager** 978.465.2351, dlyman@fawcette.com
- **Janice Mahler, Northwestern Regional Sales Manager** 949.387.6684, jmahler@fawcette.com
- **Cecile Pendleton, Account Manager** 650.833.7120, cpendleton@fawcette.com

This listing is provided as a courtesy to our readers and advertisers. The publisher assumes no responsibility for errors or omissions.
Sales Department: 209 Hamilton Avenue • Palo Alto, CA, USA 94301-2500 • 650-833-7100

Casting the Last Stone

by Sean Gallagher
Editorial Director
sgallagher@fawcette.com

News travels fast in this wired and wireless world, but it still takes just as long to process. I was sitting at a recent FTP editorial offsite meeting when *Java Pro* Editor in Chief Steve Gillmor showed me a message on his Blackberry e-mail pager: Hewlettt-Packard's announcement of its acquisition of Java application server vendor Bluestone. While the importance of the announcement was clear to me then, it took a little while to comprehend exactly what it would mean to Java developers.

Now that I've parsed through it, I see the HP–Bluestone deal for what it is: just another signal of how the Java market has changed in the last year. Bluestone joins a long list of independent Java server vendors that were acquired, merged with, or otherwise assimilated in 2000. This trend is just one of the symptoms of a maturing software market for Java—and it could threaten the level of control Java developers have over their choice of deployment platform.

There's a simple reason for the consolidation. Competition for the big corporate sales has become increasingly a battle of the behemoths—IBM and BEA dominate the market, with a variety of other players jockeying for position. The greater standardization brought by J2EE may make for a level playing field, but it also creates a market where the differentiation comes increasingly from the sales pitch and not the product.

In a new software market, sales happen on the shop floor level, and companies can often win sales based on their technical strength alone. But as a technology—and the market—matures the complexion changes.

When I spoke with Bluestone Vice President John Capobianco shortly after the sale was announced, he hinted at the problem—Bluestone wasn't getting beaten in the IT shop, but "on the golf course," as he put it. The "relationship sell" has come to Java.

The deal with Hewlett-Packard will certainly help Bluestone with the golf-course sale—HP's corporate clout should at least get Bluestone to the putting green. And Bluestone will apparently become the focus of HP's lagging enterprise middleware efforts, giving HP's entire enterprise software business a much-needed shot in the arm.

While this trend is good for the big players in the market, the impact on professional Java developers is somewhat mixed. The consolidation of vendors and the top-end focus of their sales efforts means that corporate developers will have to work harder to get a seat at the table when it comes time to make a platform choice, or risk having a solution rammed down their throat from above.

If you're an independent or an ISV, the evaporation of many of the smaller Java server vendors means you'll have fewer options when bidding on turnkey projects. And you may increasingly find yourself competing with the professional services arm of an IBM, an HP, or a BEA—not to mention Sun/iPlanet.

The HP–Bluestone deal is just another signal of how the Java market has changed in the last year

On the other hand, the entry of HP into the Java business gives developers a bit more leverage in pushing for adoption of Java at the enterprise level. When the candidates for a platform purchase come from a field including IBM, Sun, Oracle, and HP, the choice might be a bit more palatable to brand-conscious CEOs and CIOs looking for a way to cover their assets.

The mid-size players who remain in the market now face something of a challenge—they'll be forced to find strategic partners to sell them into bigger accounts. From a developer's perspective, that means, hopefully, better cross-product integration.

Others will try to find specific niches to exploit. Allaire, for example, seems to be going for the small to mid-size market, opting for a Microsoft-like packaging and pricing scheme—which could do a lot to grow the Java server developer talent pool as Allaire migrates its Web-authoring audience over to JRun.

Of course, a few will end up as roadkill—yet another reason to keep your code well documented and close to standards. ■

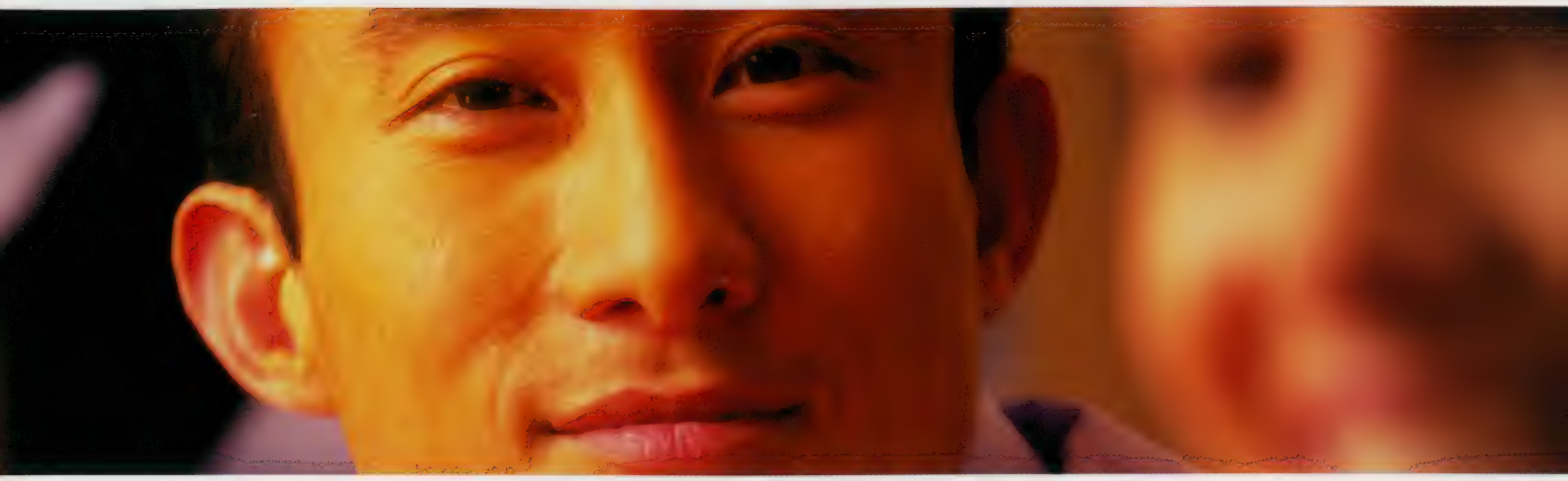
Go Online

Use the following DevX Locator+ code at www.java-pro.com to go directly to this related resource.

JP0101SG_T Read this feature online. DevX Premier Club membership is required.

Not a member yet? Go to www.DevX.com to subscribe to the Premier Club.

This IT Manager is deploying a Java™ application to 3200 users in 7 countries.



Don't blink. You might miss it.

DeployDirector™

Java application deployment and management solution.

DeployDirector is a highly scalable solution for managing the deployment and updating of client-side Java applications and applets. Now you can streamline enterprise deployment, manage version usage, and save precious IT resources. All in record time. Talk to Sitraka Software today about the advantages of DeployDirector.

For more information visit:

www.sitraka.com/deploy

Formerly KL Group 1 888-361-3264 | Europe +31(0)20 510 67 00

sitraka software

the Java™ advantage

KL Group is now Sitraka Software

A signal of our new and strengthened commitment to the Java™ marketplace, KL Group has changed its name to Sitraka Software. What can you expect from Sitraka? Innovative, highly reliable solutions. Best-of-breed products. Outstanding professional services. And the experience and knowledge to ensure you reap the full advantage of Java technology in the enterprise environment.

Makers of: **JProbe™** **JClass™** **DeployDirector™**

sitraka
software

the Java™ advantage

www.sitraka.com

1 888-361-3264 | Europe +31 (0)20 510 67 00

© Copyright 2000 By Sitraka Inc. All rights reserved. Sitraka, JProbe and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. All other products are trademarks or registered trademarks of their respective companies.

JAVAPro

YES ☑ Send me one year (13 issues) of *Java™ Pro* for only $29.95

Name _____

Address _____

City _____ State _____

Zip _____ Country _____

E-mail address _____

❑ Bill me ❑ Payment enclosed Charge my: ❑ Visa ❑ Master Card ❑ AMEX

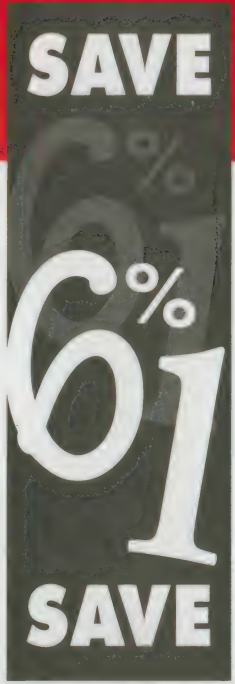
Card # _____

Expiration date _____

Signature _____

One year is 12 issues plus one special issue. Savings based on annual cover price of $77.35. Please allow 4-6 weeks for delivery of your first issue. International subscriptions must be payable in US dollars plus postage: Canada/Mexico add $18/year for surface delivery; all other countries add $44/year for airmail delivery. Canadian GST included.

4186



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 317 Palo Alto, CA

POSTAGE WILL BE PAID BY ADDRESSEE

JAVA™Pro

P.O. Box 54584
Boulder, CO 80323-4584

JAVA™ Pro Savings Voucher

Get Java™ Pro for only $2.30 per issue—and save over 61% off the cover price. A single tip in one issue can more than pay for a year's subscription. Order Now and Save!....

Annual Cover Price	Your Price	Subscription Savings
$77.35	$29.95	61%

❑ **1 Year** (13 issues) $29.95—SAVE 61%

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

E-MAIL ADDRESS_____

❑ Payment enclosed ❑ Bill me later

My Guarantee:
If at any time I do not agree that *Java™ Pro* is the best, most useful source of information on Java Programming, I can cancel my subscription and recieve a full refund.

One year is 12 monthly issues and one special issue. Savings based on annual cover price of $77.35. Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in U.S. dollars plus postage. Canada and Mexico add $18/year, all other countries add $44/year for airmail delivery. Canadian GST included.

4185

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

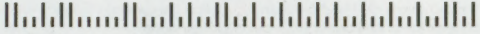
BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 317 Palo Alto, CA

POSTAGE WILL BE PAID BY ADDRESSEE

JAVA™ Pro

P.O. Box 54584
Boulder, CO 80323-4584



PASS IT ON.

☐ I want to learn how to control the synchronization between code and models.

☐ Please send me the FREE JAVA white paper and CD.

Request yours at www.rational.com/javapro or complete and mail this postage-paid reply card.

NAME _____ TITLE _____

COMPANY _____

ADDRESS _____

CITY/STATE/ZIP _____

PHONE _____ FAX _____

EMAIL _____

AR-JAVA2-Do

Rational®
the e-development company™

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 9556 LEXINGTON MA

POSTAGE WILL BE PAID BY ADDRESSEE

RATIONAL SOFTWARE
20 MAGUIRE ROAD
LEXINGTON MA 02421-9566